UCIe Standard Package
Physical Layer

Design and Verification

Based on UCIe Specification Rev 3.0 Ver 1.0

Prepared By:

Abdallah Mohamed Salah Elden

Abdelrahman Adwe Ali

Mohamed Adel

Mohamed Anwar Abd Allatif Mohamed

Mohamed Elsayed Ebrahim Mohamed

Momen Mostafa Mohamed Elzaghawy

Sondoss Mohammad Hassan

Yousef Mohamed Abdelfattah Ibrahim

Supervisor: Dr. Ali A. Al-Hamid

Al-Azaher University

Faculty of Engineering

Department of Electrical Engineering

Electronics and Communications Engineering

July 6, 2026

Abstract

The rapid proliferation of chiplet-based, disaggregated System-in-Package (SiP) architectures has created a demand for open, standardized die-to-die interconnects capable of delivering high bandwidth, low latency, and interoperability across heterogeneous silicon. The Universal Chiplet Interconnect Express (UCIe) standard addresses this need by defining a complete layered communication stack for chiplet-based systems. This thesis presents the digital hardware architecture and implementation of a UCIe Standard Package Physical Layer (PHY) compliant with UCIe Specification Revision 3.0.

The scope of this work is restricted to the digital domain. Analog components — including Phase-Locked Loops (PLLs), Analog Front-End (AFE) circuits, Phase Interpolators (PIs), and Voltage Reference (Vref) DACs — are abstracted through their digital control and status interfaces, yielding a fully synthesizable design independent of any specific analog IP technology.

The implemented architecture is organized into five principal subsystems. The Mainband datapath implements data transmission, scrambling, lane mapping, lane reversal, width degradation, and training support functions for both x16 and x8 operating widths. The Sideband subsystem provides the complete serial communication interface, including packet routing, register access transactions, Sideband Feature Extensions (SFES), and Sideband Performant Mode Operation (PMO). A comprehensive Link Training State Machine (LTSM) covers all UCIe-defined initialization, calibration, training, retraining, and error recovery procedures across all eight top-level states and thirteen mainband training sub-states. The Raw Die-to-Die Interface (RDI) state machine provides the standardized boundary between the PHY and the Die-to-Die Adapter, while a standards-compliant Register File provides configuration, status monitoring, training control, error logging, and runtime link management.

The implemented architecture was verified using a hierarchical simulation methodology covering all five subsystems, demonstrating correct behavioral operation across link initialization, training, and active-state scenarios.

The design excludes Priority Sideband Packet Transfer (PSPT), L2 Sideband Power Down (L2SPD), Tx Adjustment during Runtime Recalibration (TARR), PHY Compliance Testing support, Multi-module Configuration, and vendor-defined extensions. All architectural decisions are derived strictly from the normative requirements of UCIe Specification Revision 3.0. To the authors' knowledge, this work represents one of the first openly documented, specification-driven digital implementations of the UCIe Standard Package PHY based on Revision 3.0, yielding a modular, verifiable architecture suitable for deployment as a reusable PHY IP block in heterogeneous chiplet-based System-in-Package designs.

Acknowledgment

First and foremost, we express our deepest gratitude to Allah for granting us the strength, patience, and perseverance to successfully complete this project.

We would like to extend our sincere appreciation to our supervisor, Dr. Ali A. Al-Hamid, for his invaluable guidance, continuous support, constructive feedback, and encouragement throughout the development of this project. His expertise and insights have played a vital role in shaping this work and enhancing our understanding of digital design and verification.

We are also deeply grateful to the faculty members and teachers of the Department of Electronics and Communications Engineering, Faculty of Engineering, Al-Azhar University, for providing us with the knowledge, skills, and academic foundation that enabled us to undertake this journey.

Our sincere thanks go to Eng Abdelrahman Atef for his exceptional guidance in helping us map out the roadmap for this project, as well as Eng. Ayman Mohamed, Eng. Aliaa fouli and Eng. Mohamed Tarek for their continuous support and assistance.

We also extend our profound appreciation to Eng. Medhat Nahedh and the team at Etsal, as well as Eng. Mahmoud Samy and at SI-Vision, for their invaluable feedbacks, mentorship, and industry insights that greatly enriched this work.

Special thanks are extended to our colleagues, friends, and everyone who contributed directly or indirectly to the completion of this project through their assistance, insightful discussions, and valuable support.

Finally, we dedicate this work to our families and all those who stood by us and contributed to our success.

List of Abbreviations

Abbreviation	Definition
Ack	Acknowledge
AFE	Analog Front End
BER	Bit Error Ratio
CDC	Clock Domain Crossing
CLM	Current Lane Map
CRC	Cyclic Redundancy Check
CXL	Compute Express Link
D2C	Data-to-Clock
D2D	Die-to-Die
DCC	Duty Cycle Correction
DWORD	Double Word (four bytes)
FDI	Flit-Aware Die-to-Die Interface
FIFO	First In, First Out
Flit	Link Layer Unit of Transfer
FSM	Finite State Machine
FW-CLK	Forwarded Clock over the UCIe Link for Mainband Data Lanes
HDL	Hardware Description Language
L2SPD	L2 Sideband Power Down
LCLK	Logical PHY / Adapter / RDI Operating Clock
LFSR	Linear Feedback Shift Register
LSB	Least Significant Bit

Abbreviation	Definition
LSM	Adapter Link State Machine
LTSM	Link Training State Machine
MB	Mainband
MBINIT	Mainband Initialization (LTSM state)
MBTRAIN	Mainband Training (LTSM state)
MSB	Most Significant Bit
Nak	Negative Acknowledge
NOP	No Operation
PCIe	Peripheral Component Interconnect Express
PHY	Physical Layer
PLL	Phase-Locked Loop
PMO	Sideband Performant Mode Operation
PSPT	Priority Sideband Packet Transfer
QWORD	Quad Word (eight bytes)
RDI	Raw Die-to-Die Interface
RTL	Register-Transfer Level
Rx	Receiver
SB	Sideband
SBINIT	Sideband Initialization (LTSM state)
SiP	System in Package
SM	State Machine
SoC	System on a Chip

Abbreviation	Definition
TARR	Tx Adjustment during Runtime Recalibration
Tx	Transmitter
UCIe	Universal Chiplet Interconnect Express
UCIe-S	x16 Standard Package Module
UI	Unit Interval
vLSM	Virtual Link State Machine
Vref	Receiver Reference Voltage

Table of Contents

1.0 Introduction 21
1.1 Background and Motivation 21
1.1.1 The Shift Toward Chiplet-Based Architectures 21
1.1.2 The Need for a Standardized Die-to-Die Interconnect 22
1.1.3 UCIe as an Open Interconnect Standard 23
1.1.4 UCIe Layered Architecture 24
1.2 Related Work 26
1.2.1 Pre-UCIe Die-to-Die Interconnect Technologies 26
1.2.2 Gap in Open, Spec-Compliant Digital PHY Implementations 27
1.3 Problem Statement 28
1.4 Project Objectives 29
1.4.1 Implement the Mainband Datapath 29
1.4.2 Implement the Sideband Subsystem with PMO and SFES 29
1.4.3 Implement the Main State Machine 29
1.4.4 Implement the Register File 30
1.5 Project Scope 30
1.5.1 Inclusions 30
1.5.2 Exclusions 31
1.5.3 Analog Abstraction Boundary 31
1.6 Contributions 32
1.7 Thesis Organization 32
2.0 UCIe Standard Package PHY: Specification Analysis 34
2.1 Standard Package Module Overview 34
2.1.1 Module Signal Structure 34
2.1.2 x16 and x8 Width Configurations 36
2.1.3 Analog Abstraction Boundary 37
2.1.4 PHY Layer Position within the UCIe Stack 39
2.2 Mainband Specification 40
2.2.1 Byte-to-Lane Mapping and Valid Framing 40
2.2.2 Scrambling and Training Pattern Generation (LFSR) 44

2.2.3 Lane Reversal 47
2.2.4 Width Degradation 48
2.2.5 Training Pattern Types 50
2.3 Sideband Specification 51
2.3.1 Serial Packet Structure and Timing 51
2.3.2 Packet Types and Message Formats 53
2.3.3 Sideband Feature Extensions (SFES) 56
2.3.4 Sideband Performant Mode Operation (PMO) 58
2.4 Link Training State Machine (LTSM) Specification 59
2.4.1 State Definitions and Transition Rules 59
2.4.2 MBINIT Sub-states 62
2.4.3 MBTRAIN Sub-states 63
2.4.4 LINKINIT, ACTIVE, PHYRETRAIN, and TRAINERROR 66
2.5 Raw Die-to-Die Interface (RDI) Specification 68
2.5.1 RDI Signal List and State Machine 68
2.5.2 RDI State Transitions and PHY Control Functions 71
2.6 Register File Specification 73
2.6.1 Config Space Register Map 74
2.6.2 MMIO PHY Register Block 75
2.6.3 Register Attribute Enforcement 77
3.0 System Architecture 81
3.1 Design Philosophy and Methodology 81
3.1.1 Subsystem Decomposition Strategy 82
3.1.2 Analog Abstraction Boundary Definition 83
3.2 Top-Level Module Hierarchy 84
3.2.1 UCIe PHY as System Integration Wrapper 84
3.2.2 Four Top-Level Subsystems 85
3.2.3 Architectural Data Flow 86
3.3 Clock Domain Architecture 87
3.3.1 High-Speed MainBand Serialization Clock 87
3.3.2 MainBand & MainSM Functional Clock 88
3.3.3 High-Speed SideBand Serialization Clock 89

3.3.4 SideBand Functional Clock 90
3.4 Inter-Subsystem Interfaces 90
3.4.1 MainSM ↔ Sideband 90
3.4.2 Sideband ↔ Reg File Register Access Bus 91
3.4.3 MainSM ↔ Mainband 92
3.4.4 Reg File ↔ MainSM 92
3.5 Top-Level Parameters and Capability Straps 93
3.5.1 Configurable Parameters 93
3.5.2 Static Capability Straps 94
4.0 Mainband Design 96
4.1 Subsystem Overview 96
4.1.1 MainBand Top-Level Structure 96
4.2 Transmit (TX) 97
4.2.1 Mapper: 98
4.2.2 LFSR TX 100
4.2.3 Valid Lane Transmitter 103
4.2.1 Lane Reversal 104
4.2.2 Clock Pattern Generator 104
4.2.3 Data Lane Serializer 106
4.2.4 Valid Lane Serializer 107
4.2.5 Clk Gate 108
4.3 Receive (RX) 109
4.3.1 Valid Lane Deserializer 109
4.3.2 Date Lane Deserializer 111
4.3.3 Valid Comparator 113
4.3.4 Clock Pattern Detector 114
4.3.5 LFSR RX 116
4.3.6 Pattern Comparator 118
4.3.7 Demapper 120
5.0 Sideband Design 122
5.1 Subsystem Overview 122
5.1.1 SideBand Internal Module Hierarchy 122

5.2 Sideband SerDes 124
5.2.1 SB Serializer 124
5.2.2 PMO Enable Path 125
5.2.3 SB De-serializer 125
5.3 Link Controller 127
5.3.1 TX Priority Arbiter 127
5.3.2 SB Mapper 128
5.3.3 SB Demapper 129
5.3.4 Link Demux 130
5.3.5 SB Pattern Generation and Detection Mode 131
5.4 Training Management 133
5.4.1 TX Packetization 133
5.4.2 Link DePacketizer 134
5.4.3 RR Arbiter 136
5.4.4 Demux 137
5.4.5 Asynchronous FIFOs 137
5.5 RDI Controller 138
5.5.1 Downstream Path 139
5.5.2 Upstream Path 140
5.6 Register Access 140
5.6.1 Reg Depacketizer 141
5.6.2 Reg Access FSM 142
5.6.3 Register File Bus Interface 144
5.6.4 Completion Generator 145
6.0 MainSM Design 147
6.1 MainSM Wrapper Overview 147
6.2 LTSM 147
6.2.1 LTSM Architecture 147
6.2.2 LTSM Controller 149
6.2.3 RESET 151
6.2.4 SBINIT 152
6.2.5 MBINIT 154

6.2.6 MBTRAIN 167
6.2.7 LINKINIT 192
6.2.8 ACTIVE 192
6.2.9 L1_L2 193
6.2.10 PHYRETRAIN 195
6.2.11 D2C Shared Sweep Engine 195
6.2.12 TRAINERROR & TRAINERROR Handshake 196
6.3 RDI SM Design 197
6.3.1 RDI SM Architecture 197
6.3.2 State Machine 197
6.3.3 Handshake logic 199
6.3.4 Gating Logic 209
6.3.5 Signal Transition Detector 210
6.3.6 Status Decoder 212
6.3.7 Message handler 215
7.0 Register File Design 218
7.1 Subsystem Overview 218
7.1.1 Role within the UCIe PHY System 218
7.1.2 Clock Domain and Sideband Access Path (clk_sb via Reg_Access) 218
7.2 Address Space Architecture 219
7.2.1 25-bit Address Bus Encoding 219
7.2.2 Config Space Window (addr[24]=0, 000h–023h) 219
7.2.3 MMIO PHY Block Window (addr[24]=1, 1000h–1108h) 219
7.2.4 Address Decode Logic (cfg_sel, phy_mmio_sel) and UR Error (addr_err_o) 219
7.3 Config Space Register Map 220
7.3.1 PCIe Extended Capability Header and DVSEC Headers (000h–00Ah) 220
7.3.2 UCIe Link Capability (00Ch) — RO + HWInit 220
7.3.3 UCIe Link Control (010h) — RW + HWInit 220
7.3.4 UCIe Link Status (014h) — RO + RW1C + RW1CS (Mixed Attributes) 220
7.3.5 Link Event and Error Notification Control (018h–01Ah) 221
7.3.6 Register Locator (01Ch–020h) 221
7.4 MMIO PHY Register Block 221

7.4.1 PHY Capability (1000h) — RO + HWInit 221
7.4.2 PHY Control (1004h) — RW 222
7.4.3 PHY Status (1008h) — RO Live HW Inputs 222
7.4.4 PHY Initialization and Debug (100Ch) — RW + RO 222
7.4.5 Training Setup Registers (1010h–1050h) — RW 222
7.4.6 Current Lane Map Module 0 (1060h) — RW 223
7.4.7 Error Log 0 (1080h) — ROS State Capture Shift Register (N, N-1, N-2) 223
7.4.8 Error Log 1 (1090h) — ROS + RW1CS (Mixed) and State N-3 223
7.4.9 Runtime Link Test Control and Status (1100h–1108h) 224
7.5 Register Attribute Enforcement 224
7.5.1 RO — Hardwired Constants and Write-Ignore 224
7.5.2 HWInit — Capability Strap Sampling at Reset, Held RO Afterwards 224
7.5.3 RW — Byte-Enable Driven SW Write Path 224
7.5.4 RW1C and RW1CS — HW Set Path, SW Write-1-to-Clear, Sticky Behavior 225
7.5.5 ROS — Error-Event Triggered Single Write, Reset-Only Clear 225
7.6 Read and Write Datapath 225
7.6.1 Byte-Granular Write Path (cfg_mem / mmio_mem byte arrays, rf_be enforcement) 225
7.6.2 Registered Read Path (1-cycle latency, rdata_vld pulse) 225
7.6.3 32-bit vs 64-bit Access (rf_is_64b_access) 226
7.7 HW Input and Control Output Interfaces 226
7.7.1 Capability Strap Inputs → HWInit Fields (Link Capability, PHY Capability) 226
7.7.2 Live HW Status Inputs → RO Fields (PHY Status, Link Status, Runtime Test Status) 226
7.7.3 Error Event Inputs → ROS Capture (err_capture_en, err_state_capture, Error Logs) 227
7.7.4 SW Control Outputs (*_ctrl_out) → MainSM and Mainband 227
8.0 Simulation, Verification and Hardware Validation 228
8.1 Verification Strategy Overview 228
8.1.1 Verification Levels (Unit, Wrapper, Integration, System) 228
8.1.2 Simulation Environment (QuestaSim, Makefile/PowerShell Infrastructure) 229
8.1.3 Self-Checking Testbench Methodology (Pass/Fail Counters, Watchdog Timers) 229
8.2 Mainband Verification 229
8.2.1 SER/DES Loopback Wrapper with Frame-Lock Scoreboard (unit_tx_deser_wrapper_tb) 230
8.2.2 Full TX→RX Loopback with Demapper Verification (unit_tx_demap_wrapper) 230

8.2.3 Framed Path and Stream Path Integration (unit_mb_framed_path_tb, unit_mb_path_stream_tb) 230
8.2.4 Die-to-Die Integration Testbench with Fault Injection (unit_mb_die2die_tb) 230
8.2.5 Full Mainband Training Sequence Verification (unit_mb_train_seq_tb) 231
8.3 Sideband Verification 231
8.3.1 sb_serializer SVA Formal Properties (packet length, gap enforcement, clock forwarding) 231
8.3.2 sb_deserializer SVA Formal Properties (counter wrap, packet-done position, CDC pulse) 231
8.3.3 PMO Mode Behavioral Verification 232
8.4 LTSM and D2C Verification 232
8.4.1 Unit-Level D2C Point Test Testbenches (unit_RX_D2C_PT_tb, unit_TX_D2C_PT_tb) 232
8.4.2 D2C Point Test Wrapper Verification (wrapper_D2C_PT_tb, wrapper_D2C_PT_top_tb — 100 Randomized Iterations) 232
8.4.3 MBTRAIN Sub-state Directed Testbenches (all 13 sub-states) 233
8.4.4 MBTRAIN Top-Level Integration Testbench (wrapper_MBTRAIN_tb) 233
8.4.5 Class-Based OOP MBTRAIN Testbench Suite 233
8.5 RDI_SM Verification 235
8.5.1 unit_Timer_tb — 1µs and 16ms Timer Verification 235
8.5.2 RDI State Sub-module Directed Testbenches 235
8.5.3 Stall Handshake and Clock Gating Behavioral Verification 235
8.6 Register File Verification 236
8.6.1 Attribute Enforcement Testing (RO write-ignore, RW1C clear, ROS capture) 236
8.6.2 Address Decode and UR Error Response Testing 236
8.6.3 64-bit vs 32-bit Access and Byte-Enable Verification 236
8.7 System-Level Verification of the UCIe PHY 237
8.7.1 Purpose and scope 237
8.7.2 Two-die "silicon-realistic" topology 237
8.7.3 The channel model — where faults are injected 238
8.7.4 Configuration through the real register path 238
8.7.5 Driving the RDI (adapter) face 238
8.7.6 Checking — scoreboard, data proof, and observability 239
8.7.7 Making long timeouts simulatable 239
8.7.8 Regression control 239

8.7.9 Coverage-of-intent — what the 29 scenarios span 240
8.8 Verification Results Summary 241
8.8.1 Test Completion Status per Subsystem 241
8.9 FPGA Prototype and Hardware Validation 243
8.9.1 Motivation and approach 243
8.9.2 FPGA-target RTL 243
8.9.3 Clocking and constraints 243
8.9.4 Embedded system 244
8.9.5 Bare-metal application and driver 244
8.9.6 Results 245
9.0 Conclusion 246
9.1 Summary of Work 246
9.1.1 Design Objectives Revisited 246
9.1.2 Summary of Each Subsystem Implementation 246
9.2 Contributions 247
9.2.1 Spec-Compliant LTSM Covering All Normative States and Sub-states 247
9.2.2 Mainband Datapath with x8 Width Mode, Width Degradation, and Lane Reversal 248
9.2.3 Sideband Subsystem with PMO and SFES 248
9.2.4 RDI State Machine with Full Clock Gating and Stall Handshake 248
9.2.5 Register File with Full Attribute Enforcement 248
9.2.6 Multi-Level Verification Infrastructure 249
9.2.7 FPGA Prototype Demonstrating Synthesis Feasibility at 125 MHz 249
9.3 Limitations 249
9.3.1 Excluded Features (PSPT, L2SPD, TARR, Compliance Testing, Multi-module) 249
9.3.2 Analog Components Not Modeled (PLL, AFE, PI, Vref DAC) 249
9.3.3 Single-Module Scope Only 250
9.4 Future Work 250
9.4.1 Advanced Package Variant Extension (x64/x32, Lane Repair, Redundant Sideband) 250
9.4.2 Multi-module Configuration Support 250
9.4.3 PSPT and L2SPD Feature Integration 250
9.4.4 Full-System Co-simulation with a D2D Adapter BFM 251
9.4.5 ASIC Synthesis and Timing Closure 251

9.4.6 UCIe-3D Package Variant Exploration..... 251

List of Figures

Figure 1-1: Chiplet Approach 21
Figure 1-2: A Package Composed of CPU Dies, Accelerator Die(s), and I/O Tile Die Connected through UCIe 23
Figure 1-3: UCIe Layers and functionalities 24
Figure 2-1: Standard Package interface 34
Figure 2-2: Single module configuration: Standard Package 35
Figure 2-3: x16 or x8 Standard Package Module 36
Figure 2-4: Byte map for x16 interface 40
Figure 2-5: Byte to Lane mapping for Standard package x16 degraded to x8 42
Figure 2-6: Valid framing example 43
Figure 2-7: LFSR implementation 44
Figure 2-8: Test and training logic 46
Figure 2-9: Lane failure detection 46
Figure 2-10: All Lane error detection 47
Figure 2-11: Sideband Packet Transmission: Back-to-Back 51
Figure 2-12: Example 64-bit Sideband Serial Packet Transfer 51
Figure 2-13: Format for Register Access Request 54
Figure 2-14: Format for Register Access Completions 54
Figure 2-15: Messages without Data 55
Figure 2-16: Format for Messages with data payloads 55
Figure 3-1: Top Level Module Hierarchy 84
Figure 4-1: MainBand Top-Level Structure 96
Figure 4-2: Mapper Block Diagram 98
Figure 4-: Byte to Lane mapping for Standard package x16 degraded to x8 99
Figure 4-: Byte to Lane mapping for Standard package x16 99
Figure 4-5: LFSR Tx diagram 100
Figure 4-6: Valid Tx daigram 103
Figure 4-7: MB Reversal diagram 104
Figure 4-8: CLK Pattern Generator 105
Figure 4-9: Data Lane Serializer Diagram 106
Figure 4-10: Valid Lane Serializer Diagram 107
Figure 4-11: CLK Gate Diagram 108
Figure 4-12: Valid Deserializer Block Diagram 110
Figure 4-13: Data Deserializer 112
Figure 4-14: Valid Comparator 113
Figure 4-15: CLK Pattern Detector 114
Figure 4-16: LFSR Rx 116
Figure 4-17: Pattern Comparator 118
Figure 4-18: Demapper 120

Figure 5-1: SideBand Hierarchy Architecture 123
Figure 5-2: SB Serializer 124
Figure 5-3: SB Deserializer 126
Figure 5-4: Link Control Diagram 127
Figure 5-5: Tx Priority Arbiter 127
Figure 5-6: SB Mapper 128
Figure 5-7: SB Demapper 129
Figure 5-8: Link Demux 130
Figure 5-9: SB Pattern Generation and Detection 131
Figure 5-10: Train Management 133
Figure 5-11: TX Packetizer Block Diagram 134
Figure 5-12: Link DePacketizer Block Diagram 135
Figure 5-13: Round and Roben Arbiter Diagram 136
Figure 5-14: RDI Control diagram 139
Figure 5-15: Reg Access block diagram 140
Figure 5-16: Reg Depacketizer Diagram 142
Figure 5-17: FSM Diagram 143
Figure 5-18: Completion Generator Diagram 145
Figure 6-1: LTSM Block Diagram 148
Figure 6-2: LTSM State diagram 149
Figure 6-3: SBINIT state diagram 153
Figure 6-4: MBINIT Architecture 155
Figure 6-5: MBINIT.PARAM state diagram 158
Figure 6-6: MBINIT.CAL state diagram 159
Figure 6-7: MBINIT.REPAIRCLK state diagram 161
Figure 6-8: MBINIT.REPAIRVAL state diagram 162
Figure 6-9: MBINIT.REVERSALMB state diagram 164
Figure 6-10: MBINIT.REAIRMB state diagram 166
Figure 6-11: MBTRAIN Architecture 168
Figure 6-12: MBTRAIN state Diagram 170
Figure 6-13: MBTAIN.VALVREV state diagram 172
Figure 6-14: MBTRAIN.DATAVREF state diagram 174
Figure 6-15: MBTRAIN.SPEEDIDLE state diagram 176
Figure 6-16: MBTRAIN.TXSELFCAL state diagram 177
Figure 6-17: MBTARIN.RXCLKCAL state diagram 179
Figure 6-18: MBTRAIN.VALTRAINCENTER state diagram 180
Figure 6-19: MBTRAIN.VALTRAINVREF state diagram 181
Figure 6-20: MBTRAIN.DATATRAINCENTER1 183
Figure 6-21: DATATRAINVREF state diagram 184
Figure 6-22: MBTRAIN.RXDESKEW Local state diagram 187
Figure 6-23: MBTRAIN.RXDESKEW Partner state diagram 188
Figure 6-24: MBTRAIN.REPAIR state diagram 191

Figure 6-25: RDI SM Architecture 197
Figure 6-26: RDI State Machine 199
Figure 6-27: Handshake Logic Architecture 200
Figure 6-28: AWAKE Handshake State Diagram 203
Figure 6-29: CLK Handshake State Diagram 204
Figure 6-30: Stall Handshake State Diagram 206
Figure 6-31: Active Handshake State Diagram 209
Figure 6-32: Gating Logic State Diagram 210
Figure 6-33: Signal Transition Detector State Diagram 212
Figure 6-34: State Decoder diagram 214
Figure 6-35: Message Handler State machine 215
Figure 6-36: Message Handler diagram 217
Figure 8-1: UCIe system level TB Architecture 237
Figure 8-2: Vivado IP-Intergrator block design 244

List of Tables

Table 2-1: Per-lane LFSR seed values 45
Table 2-2: Standard Package logical lane map encodings 49
Table 2-3: Sideband packet opcode encodings 53
Table 2-4: Sideband routing identifier scope by interface domain 56
Table 2-5: SBFE feature bits in {MBINIT.PARAM SBFE req/resp} 57
Table 2-6: LTSM state definitions 60
Table 2-7: MBTRAIN substate sequence and objectives 64
Table 2-8: RDI control and status signal list 68
Table 2-9: Requests Considered in Each State by Lower Layer 71
Table 2-10: Register attributes 77
Table 3-1: Forward Clock Frequency and phase 88
Table 4-1: Mapper Interface Signals 100
Table 4-2: LFSR Tx Interface Signals 102
Table 4-3: Valid Tx Interface signals 103
Table 4-4: Lane Reversal Interface Signals 104
Table 4-5: Clock Pattern Generator Interface Signals 105
Table 4-6: Data Lane Serializer Interface Signals 106
Table 4-7: Valid Lane Serializer Interface Signal 107
Table 4-8: Clk Gate Interface Signals 108
Table 4-9: Valid Desierializer Interface signal 111
Table 4-10: Data Desierializer Interface signal 112
Table 4-11: Valid Comparator Interface signal 114
Table 4-12: Clk Pattern Detector Interface signals 115
Table 4-13: LFSR Rx Interface Signal 117
Table 4-14: Pattern Comparator Interface signal 119
Table 4-15: Demapper Interface signals 121
Table 5-1: SB serializer Interface Signals 125
Table 5-2: SB Deserializer Interface signals 126
Table 5-3: TX Priority Arbiter Interface Singals 128
Table 5-4: SB Mapper Interface signals 129
Table 5-5: SB Demapper Iterface signals 130
Table 5-6: Link Demux Interface signals 131
Table 5-7: SB Pattern Generation and Detection Mode Interface signals 132
Table 5-8: TX Packetization Interface signals 134
Table 5-9: Link DePacketizer Interface signals 135
Table 5-10: RR Arbiter Interface signal 136
Table 5-11: Demux Interface signals 137
Table 5-12: Asynchronous FIFOs Interface singals 138
Table 5-13: Register Access Interface signals 141

Table 5-14: Reg Depacketizer Interface signals 142
Table 5-15: Reg Access FSM Interface signals 144
Table 5-16: Register File Bus Interface signals 145
Table 5-17: Completion Generator Interface signals 146
Table 6-1: LTSM Controller Interface Signals 150
Table 6-2: RESET interface signals 152
Table 6-3: SBINIT Interface signals 154
Table 6-4: MBINIT Interface signals 155
Table 6-5: MBINIT.PARAM Interface signals 157
Table 6-6: MBINIT.CAL Interface Signals 158
Table 6-7: MBINIT.REPAIRCLK Interface Signals 160
Table 6-8: MBINIT.REPAIRVAL Interface Signals 163
Table 6-9: MBINIT.REVERSALMB Interface Signals 165
Table 6-10: MBINIT.REPAIRMB Interface Signals 167
Table 6-11: MBTRAIN Control/Result Plane Overview 168
Table 6-12: MBTRAIN substates state diagram 170
Table 6-13: MBTRAIN.VALVREF Interface Signals 171
Table 6-14: MBTRAIN.DATAVREF Interface Signals 173
Table 6-15: MBTRAIN.SPEEDIDLE Interface Signals 175
Table 6-16: MBTRAIN.TXSELFCAL Interface Signals 176
Table 6-17: MBTRAIN.RXCLKCAL Interface Signals 178
Table 6-18: MBTRAIN.VALTRAINCENTER Interface Signals 180
Table 6-19: MBTRAIN.VALTRAINVREF Interface Signals 182
Table 6-20: MBTRAIN.DATATRAINCENTER1 Interface Signals 182
Table 6-21: MBTRAIN.DATATRAINVREF Interface Signals 185
Table 6-22: MBTRAIN.RXDESKEW Interface Signals 185
Table 6-23: MBTRAIN.DATATRAINCENTER2 Interface Signals 188
Table 6-24: MBTRAIN.LINKSPEED Interface Signals 189
Table 6-25: MBTRAIN.REPAIR Interface Signals 190
Table 6-26: LINKINIT Interface Signals 192
Table 6-27: ACTIVE Interface Signals 193
Table 6-28: L1 Interface Signals 194
Table 6-29: L2 Interface Signals 194
Table 6-30: PHYRETRAIN Interface Signals 195
Table 6-31: TRAINERROR Interface Signals 196
Table 6-32: State Machine Interface signals 198
Table 6-33: Handshake Logic Interface signals 201
Table 6-34: awake Handshake Interface Signals 202
Table 6-35: clk_handshake Interface Signals 204
Table 6-36: stall handshake Interface Signals 205
Table 6-37: active handshake Interface Signals 208
Table 6-38: Gating Logic Interface Siganls 210

Table 6-39: Signal Transition Detector Interface signals 211
Table 6-40: Status Decoder Interface signals 213
Table 6-41: Message Handler Interface signals 216
Table 8-1: The four layers verification 228
Table 8-2: Types of Injected Error 238
Table 8-3: The 29 system-level scenarios 240
Table 8-4: Test Completion Status 241
Table 8-5: Clock Domains 243
Table 8-6: Embedded System Hardware Channels (UCIe PHY) 244

1.0 Introduction

1.1 Background and Motivation

1.1.1 The Shift Toward Chiplet-Based Architectures

For several decades, the semiconductor industry relied primarily on monolithic System-on-Chip (SoC) integration, in which processor cores, memory controllers, input/output interfaces, accelerators, and other functional blocks were fabricated together on a single silicon die. This approach benefited directly from Moore's Law, as successive process generations provided higher transistor densities, improved performance, lower power consumption, and reduced cost per transistor. As a result, increasingly complex systems could be integrated onto a single chip while maintaining favorable manufacturing economics.

However, continued scaling of monolithic designs has become increasingly challenging as semiconductor technologies advance into deep submicron process nodes. Manufacturing constraints, including photolithographic reticle limits, impose practical bounds on achievable die size, while fabrication yield decreases as die area increases due to the higher probability of defects. In addition, different functional blocks often exhibit distinct process requirements. High-speed analog circuits, SerDes interfaces, and embedded memories may achieve superior performance, power efficiency, or cost when implemented on mature process technologies rather than the most advanced digital nodes. Consequently, integrating all system components onto a single process technology can lead to suboptimal trade-offs in cost, performance, and design flexibility.

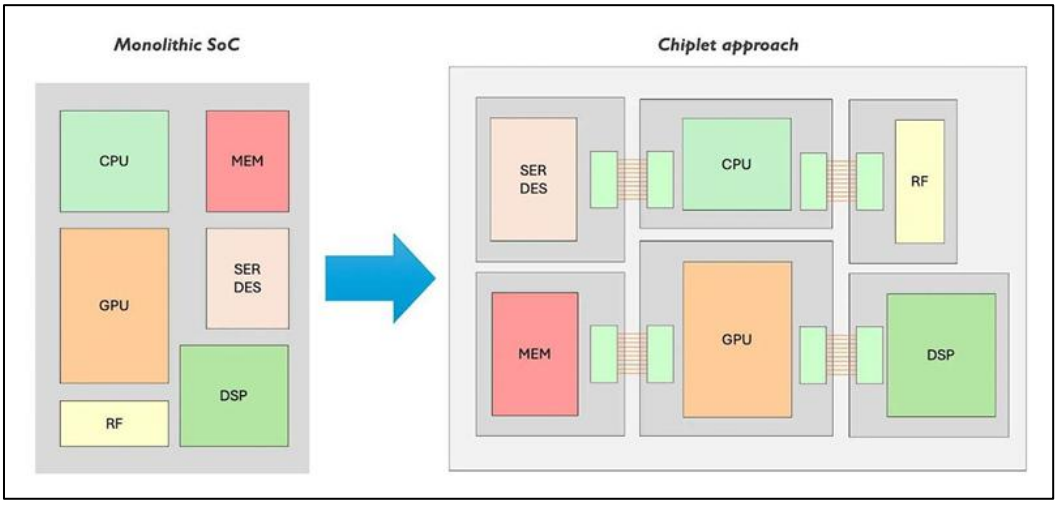


Figure 1-1: Chiplet Approach

To address these limitations, the industry has increasingly adopted chiplet-based architectures. Instead of implementing an entire system on a single die, functionality is partitioned into multiple smaller dies, known as chiplets, each optimized for a specific function and process technology.

These chiplets are subsequently integrated within a common package and interconnected through high-speed die-to-die communication links.

Modern processors and accelerators across the industry increasingly employ this approach to combine compute, memory, and I/O functions within a common package, demonstrating its viability across a broad range of application domains.

This disaggregated approach improves manufacturing yield, enables heterogeneous process-node integration, facilitates IP reuse, and allows independent development and validation of individual components. As a result, chiplet-based integration has emerged as a key strategy for sustaining system-level performance scaling and has gained widespread adoption across modern CPUs, GPUs, AI accelerators, networking devices, and high-performance computing platforms. However, the success of chiplet-based systems depends critically on the efficiency and interoperability of the die-to-die interconnect that links these chiplets together.

1.1.2 The Need for a Standardized Die-to-Die Interconnect

The transition to chiplet-based architectures introduced a fundamental engineering challenge: while disaggregation improves yield, flexibility, and process node optimization, it creates a critical dependency on the die-to-die (D2D) interconnect that links these disaggregated components on a common package. The performance, power efficiency, and latency characteristics of the overall system are now directly governed by this interconnect, elevating it from a secondary concern to a first-order design constraint.

In the absence of a unified standard, semiconductor vendors developed proprietary D2D interconnect solutions tailored to their specific chiplet ecosystems. These solutions, while optimized for intra-vendor integration, fundamentally precluded cross-vendor interoperability. A chiplet designed by one vendor could not be seamlessly integrated with a die from another vendor without costly adaptation layers or re-spins, negating many of the economic and flexibility advantages that chiplet disaggregation was intended to provide. The absence of interoperability prevented the emergence of a vendor-neutral chiplet ecosystem, creating a barrier to the formation of an open chiplet market where intellectual property (IP) blocks from independent vendors could be composed into heterogeneous System-in-Package (SiP) solutions.

Furthermore, proprietary interconnects imposed significant burdens on the IP supply chain. Each integration required custom interface design, bespoke verification environments, and point-to-point characterization, substantially increasing integration cost and time-to-market. The absence of a shared compliance and debug framework further compounded this challenge, as post-silicon validation between dies from different sources carried disproportionate overhead relative to monolithic SoC validation.

A standardized on-package interconnect was therefore essential to unlock the full potential of chiplet-based integration. Such a standard must define not only the physical and electrical interface, but also the protocol mapping, link training, initialization, and parameter negotiation mechanisms necessary to guarantee plug-and-play interoperability across a diverse ecosystem of vendors and process technologies. It must accommodate multiple communication protocols,

support a range of packaging technologies and bump pitches, and provide a well-defined compliance path to reduce integration risk.

Universal Chiplet Interconnect express (UCIe) was developed to fulfill this role. As an open, multi-protocol capable, on-package interconnect standard, UCIe provides a unified framework that enables a vibrant ecosystem of disaggregated die architectures interconnectable across vendor boundaries, with support for both Standard and Advanced packaging substrates.

1.1.3 UCIe as an Open Interconnect Standard

Universal Chiplet Interconnect Express (UCIe) is an open, multi-protocol capable, on-package interconnect standard developed to enable the construction of disaggregated, heterogeneous

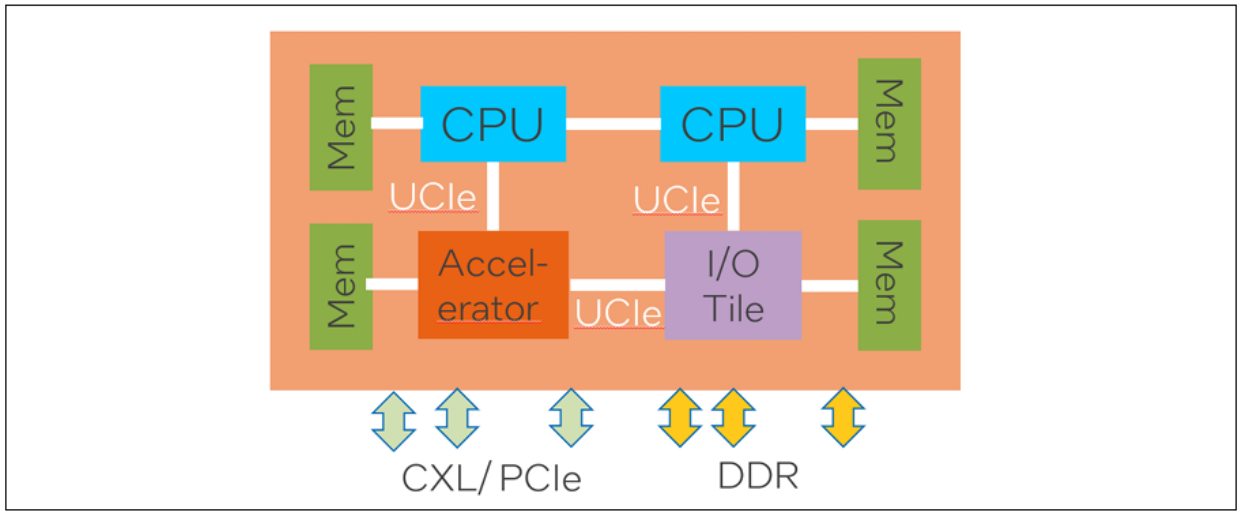


Figure 1-2: A Package Composed of CPU Dies, Accelerator Die(s), and I/O Tile Die Connected through UCIe

System-in-Package (SiP) solutions from chiplets sourced across multiple vendors and process technologies. Its defining characteristic as an open standard is the full specification of every interface boundary within the UCIe stack, enabling independent development and integration of each functional layer from any IP provider while guaranteeing interoperability through well-defined compliance mechanisms.

UCIe is organized as a three-layer hierarchy. The Protocol Layer is responsible for generating and consuming protocol Flits, with support for PCIe, CXL, Streaming, and a protocol-agnostic Raw Format. The Die-to-Die (D2D) Adapter sits between the Protocol Layer and the Physical Layer, handling reliable data transfer via CRC and Retry where required, arbitration and multiplexing of multiple protocol stacks onto a shared physical link, and coordinating Link State Management and parameter negotiation with the remote link partner. The Physical Layer is responsible for the electrical interface to the package bumps, encompassing link training, lane repair and reversal where applicable, scrambling, sideband initialization, and clock forwarding. These layers are connected through two formally defined interfaces: the Flit-aware Die-to-Die Interface (FDI)

between the Protocol Layer and the Adapter, and the Raw Die-to-Die Interface (RDI) between the Adapter and the Physical Layer.

UCIe accommodates three distinct packaging technologies to address a wide range of integration requirements. The Standard Package targets cost-sensitive applications on organic substrates with long channel reach, the Advanced Package targets performance-optimized integration with higher bump density and shorter channel reach, and UCIe-3D extends the standard to face-to-face die stacking scenarios. Together, these variants span a broad spectrum of packaging technologies, bandwidth densities, and power efficiency targets; detailed electrical and performance characteristics for each packaging variant are addressed in Chapter 2.

Interoperability across the UCIe ecosystem is guaranteed by the specification's strict definition of PHY dimensions, bump map layouts, protocol negotiation flows, and link training sequences. The specification is further designed to evolve in a backward-compatible manner, ensuring that future revisions can be adopted incrementally without invalidating prior implementations. This combination of openness, layered architecture, multi-protocol flexibility, and formally specified interoperability positions UCIe as the foundational standard for the emerging open chiplet economy.

1.1.4 UCIe Layered Architecture

The UCIe stack is organized as a well-defined three-layer hierarchy, where each layer performs a bounded set of functions, communicates with adjacent layers exclusively through formally specified interfaces, and can be independently implemented and sourced. This separation of concerns is a deliberate architectural decision, enabling IP providers to develop and verify

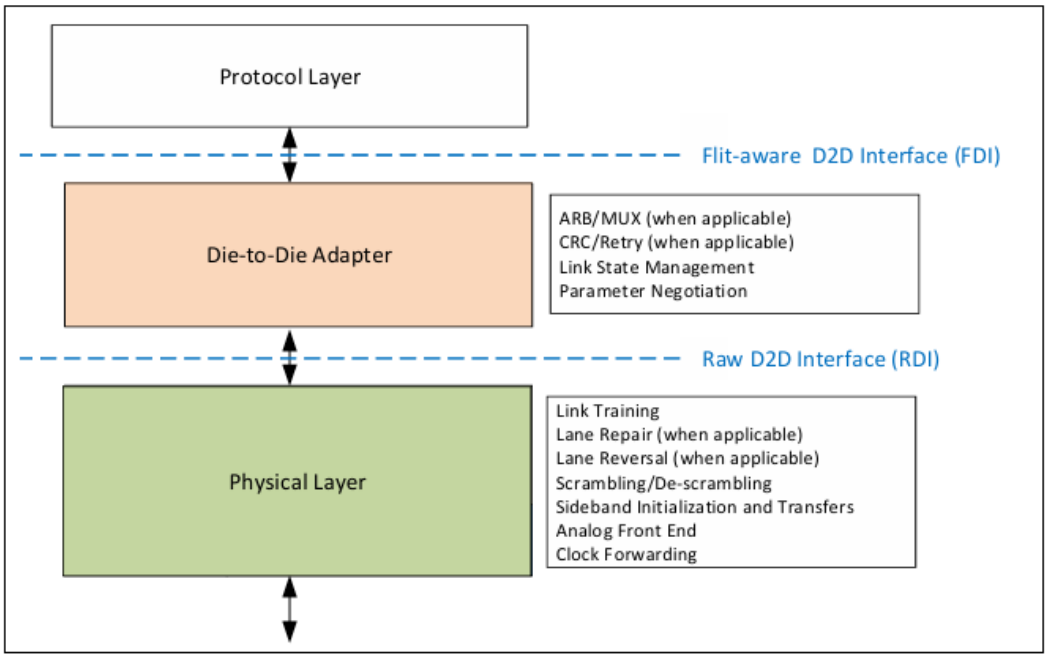


Figure 1-3: UCIe Layers and functionalities

individual layers in isolation while guaranteeing system-level interoperability through compliance to the specified interface contracts.

Protocol Layer. The topmost layer is application-specific and is responsible for generating and consuming data in a Flit-based format. UCIe supports four protocol mappings at this layer: PCIe, CXL, Streaming, and a protocol-agnostic Raw Format that permits any user-defined protocol to be transported provided both link partners support it. The Protocol Layer does not participate in link training, parameter negotiation, or error recovery; these responsibilities are delegated entirely to the layer below. Each protocol maps onto one of several defined Flit formats, which determine the structure of the data presented to the Adapter at the FDI boundary.

Die-to-Die (D2D) Adapter. The Adapter is the central coordination layer of the UCIe stack. It interfaces upward to the Protocol Layer via the Flit-aware Die-to-Die Interface (FDI) and downward to the Physical Layer via the Raw Die-to-Die Interface (RDI), serving as the boundary at which protocol-level data is mapped onto physical-layer-ready transfers.

From a reliability standpoint, the Adapter provides CRC computation and Retry for channel conditions where raw error rates exceed the threshold at which the specification mandates protected transfer. When two protocol stacks are configured to share a single physical link, it additionally performs arbitration and multiplexing to serialize their traffic onto the shared channel.

At the link coordination level, the Adapter implements the Link State Machine (LSM) and synchronizes its state transitions with the RDI state machine, while conducting parameter negotiation with the remote link partner to establish the operational Flit format, protocol mode, and power management capabilities. The Adapter is designed to minimize logic insertion on the main data path, preserving the low-latency characteristics of the interconnect.

Physical Layer. The Physical Layer is subdivided into two functional blocks. The focus of this thesis is the digital PHY Logic block. The Analog Front End (AFE), instantiated as k identical slices forming the Module, contains all analog circuitry including transmitters, receivers, and clock forwarding circuits; in the digital implementation context of this project, this block is abstracted to its digital control signal boundary. The PHY Logic block encompasses all digital control functions: sideband initialization and packet transfer, Link Training State Machine (LTSM) execution, scrambling and de-scrambling of mainband data, lane repair and reversal for Advanced Package configurations, and width degradation for Standard Package configurations. The Physical Layer's sideband interface operates on an independent auxiliary power domain at a fixed 800 MHz clock, remaining operational regardless of mainband power state.

Defined Interface Boundaries. UCIe formally specifies two internal interface contracts. The FDI governs all signal exchange between the Protocol Layer and the Adapter, including data path, state status, flow control, and sideband mailbox access. The RDI governs all signal exchange between the Adapter and the Physical Layer, including data path, link state coordination, power management handshaking, and configuration access. The RDI signal list is a proper subset of the FDI signal list by design. These two interfaces collectively enforce the layered abstraction: no layer requires visibility into the internal implementation of any other layer, and the full system behavior is determined

entirely by the specification of these interface boundaries and the layer-level behavioral rules defined in the UCIe specification.

1.2 Related Work

1.2.1 Pre-UCIe Die-to-Die Interconnect Technologies

Prior to the emergence of UCIe as an open multi-protocol standard, the chiplet interconnect space was characterized by a collection of vendor-developed or consortium-developed die-to-die (D2D) physical interfaces, each targeting specific integration requirements but sharing a common limitation: the absence of a complete, cross-vendor interoperability framework encompassing physical signaling, link training, parameter negotiation, and protocol transport.

Advanced Interface Bus (AIB). AIB was developed by Intel and subsequently open-sourced under the CHIPS Alliance as part of broader industry efforts toward disaggregated SoC construction. It defines a parallel, source-synchronous interface using a forwarded clock architecture, targeting short-reach on-package connectivity with a bump pitch compatible with organic substrate packaging. AIB 1.0 provides a 20-lane unidirectional data path with an asynchronous sideband channel multiplexed onto the main data lanes, while AIB 2.0 extends the data path to 40 lanes per direction with improved power efficiency. The interface is intentionally minimal at the physical level: it provides a raw data transport channel without defining a protocol layer, a link training state machine, or a standardized parameter negotiation flow. Initialization and configuration are left to vendor-specific implementations layered above the AIB PHY, making cross-vendor integration dependent on bilateral agreements rather than specification compliance.

Bunch of Wires (BoW). The Bunch of Wires interface was developed under the Open Compute Project (OCP) with the goal of providing a low-cost, high-bandwidth D2D interconnect solution for standard organic substrate packaging. BoW similarly employs a forwarded clock, parallel data path architecture and defines bump map layouts and electrical characteristics for a fixed set of module widths. Like AIB, BoW does not define a multi-protocol transport framework, a standardized link training sequence, or a cross-layer parameter negotiation mechanism. Power management coordination, error handling, and protocol mapping remain outside the specification boundary, leaving substantial integration burden on the SoC integrator. Although AIB and BoW continue to serve niche integration use cases, neither evolved into a complete ecosystem-level interoperability standard comparable to UCIe.

High Bandwidth Memory (HBM). Although HBM is fundamentally a memory interface rather than a general-purpose chiplet interconnect, its inclusion here is relevant because it represents the dominant approach to high-bandwidth vertical die connectivity in the pre-UCIe era, and its architectural constraints illustrate the scope limitations that motivated a more general-purpose standard. Standardized through JEDEC, HBM targets DRAM-to-logic die connectivity using wide parallel buses and through-silicon vias (TSVs), achieving exceptional bandwidth density. However, its interface is rigidly defined around a memory access model and is not extensible to arbitrary

protocol transport. It does not address the link training, parameter negotiation, or multi-protocol multiplexing requirements that general-purpose chiplet integration demands.

Shared Limitations. Despite their individual contributions, AIB, BoW, and HBM share a set of structural deficiencies that precluded their adoption as foundations for an open chiplet ecosystem. None establishes a compliance-driven interoperability framework capable of guaranteeing communication between independently developed chiplets from different vendors — arguably the most critical requirement for a viable open chiplet market. None was designed to provide a standardized multi-protocol transport framework comparable to that defined by UCIe, none defines a standardized debug infrastructure for post-silicon fault isolation across die boundaries, and none provides a unified link state management framework governing initialization, training, and power management in a vendor-agnostic manner. These gaps collectively motivated the development of UCIe as a specification addressing the complete die-to-die communication stack, from the physical package interface to protocol-layer transport, while replacing bilateral vendor agreements with a compliance-verifiable open standard.

1.2.2 Gap in Open, Spec-Compliant Digital PHY Implementations

While UCIe has attracted significant commercial interest since its initial release in 2022, the landscape of publicly available, specification-compliant digital PHY implementations remains sparse. Commercial silicon IP vendors have developed proprietary UCIe PHY hard macros targeting specific process nodes, but these implementations are closed, process-tied, and inaccessible for academic study, architectural exploration, or independent verification. To the authors' knowledge, openly documented implementations covering the full scope of the UCIe Revision 3.0 Standard Package PHY digital layer remain extremely limited, and no implementation of comparable scope was identified during the course of this work

This gap is particularly consequential at the digital PHY logic layer, which governs the entirety of link bring-up, training, and runtime management The UCIe PHY digital logic encompasses a deeply stateful Link Training State Machine (LTSM) comprising eight top-level states and thirteen MBTRAIN sub-states, a sideband protocol engine operating on an always-on auxiliary power domain, a full register file spanning Physical Layer capability, control, status, and error logging registers, an RDI state machine coordinating power management and active state transitions with the D2D Adapter, and datapath functions including scrambling, lane reversal, and width degradation. The specification-compliance burden across these subsystems is substantial: the UCIe specification defines precise sequencing rules, sideband message handshakes, timeout values, and reset behaviors that must be implemented in exact conformance to guarantee interoperability with a remote link partner.

Existing academic treatments of UCIe have largely focused on performance modeling, packaging characterization, or protocol-layer mapping, without producing a synthesizable, spec-traceable digital PHY implementation. The absence of such a reference creates a structural barrier: chiplet ecosystem participants seeking to validate UCIe integration behavior, explore architectural trade-offs in the PHY logic, or develop test environments for Adapter or Protocol Layer IP have no open baseline against which to work.

This project directly addresses that gap. The implementation presented in this thesis targets the UCIe Revision 3.0 Standard Package PHY digital layer, covering the complete LTSM sequence, sideband subsystem, RDI state machine, mainband datapath logic, and register file, with all design decisions traceable to specific sections of the UCIe Specification Rev 3.0. The scope is bounded to the x16 and x8 Standard Package configurations, with analog components abstracted to their digital control signal interfaces, yielding a digital-only implementation suitable for FPGA prototyping and ASIC synthesis flow integration.

1.3 Problem Statement

The proliferation of the UCIe standard as a cross-vendor die-to-die interconnect framework has established a detailed and rigorously defined behavioral contract at the physical layer. Compliant operation requires a digital logic subsystem that correctly implements a multi-state Link Training State Machine encompassing eight top-level states and thirteen sub-states within the mainband training phase alone, a sideband protocol engine operating on a permanently powered auxiliary domain, an RDI state machine mediating the interface to the D2D Adapter, and a structured register file governing capability advertisement, link control, and error logging. The interdependencies among these subsystems are non-trivial: sideband initialization must conclude with a precisely sequenced handshake before mainband training may commence, RDI state transitions are gated on LTSM state outputs, and lane-level functions such as scrambling, lane reversal, and width degradation must be engaged and disengaged at specification-defined points in the training sequence. Implementing this behavioral contract correctly, without deviation from the sequencing rules mandated by the specification, constitutes a non-trivial engineering problem that extends well beyond the design of any individual sub-block in isolation.

Despite the specification's precision, to the authors' knowledge no publicly available, open, spec-driven digital RTL reference exists for the UCIe Standard Package physical layer. Commercial chiplet IP is proprietary and inaccessible to academic or independent evaluation. The consequence is that hardware designers and researchers working on chiplet integration are left without a verifiable behavioral baseline against which custom implementations can be cross-checked or adapted. This gap is particularly acute for the Standard Package configuration, which trades the Advanced Package's lane-repair infrastructure for a lower-cost organic substrate implementation, introducing its own distinct set of digital control requirements, including width degradation as the sole mechanism for handling lane-level faults and a fixed x16 or x8 mainband data path with no redundancy.

This project addresses that gap by designing and structurally specifying a complete digital PHY layer for the UCIe Standard Package, grounded exclusively in UCIe Specification Revision 3.0. The scope is bounded to the digital domain: analog front-end circuitry is treated as abstracted components with well-defined digital control signal interfaces. Within that boundary, the implementation targets a complete digital realization of the UCIe Standard Package PHY within the defined project scope. The result is intended as a structurally complete, academically transparent digital PHY

implementation that can serve as a documented reference point for future derivative work in spec-compliant chiplet interconnect design.

1.4 Project Objectives

1.4.1 Implement the Mainband Datapath

The first objective is to implement a specification-compliant mainband datapath capable of transmitting and receiving data across the UCIe Standard Package physical link. The deliverable encompasses lane mapping, scrambling, training pattern generation and comparison, lane reversal, width degradation, and associated error-detection logic required to bring up and sustain the mainband data path under the conditions defined by the UCIe specification. Both x16 and x8 operating widths are supported, with the active lane configuration determined dynamically by the outcome of the link training sequence. The expected outcome is a structurally complete mainband datapath that correctly interfaces with the link training state machine on its control plane and with the D2D Adapter via the RDI on its data plane, producing behavior fully consistent with the Standard Package requirements of UCIe Specification Rev 3.0.

1.4.2 Implement the Sideband Subsystem with PMO and SFES

The second objective is to implement a specification-compliant sideband subsystem capable of managing all parameter exchange, register access, and link coordination messaging required throughout the link training and operational lifecycle. The deliverable encompasses the sideband protocol engine, packetization and depacketization logic, register access transactions, message routing, and the negotiation and runtime enforcement of PMO and SFES capabilities, all operating on a permanently powered auxiliary domain. The subsystem must correctly serialize and deserialize sideband packets, enforce the transmission rules associated with both standard and performant operation, and provide a message interface for all training-phase exchanges with the Link Training State Machine. The expected outcome is a sideband engine whose behavior is fully consistent with the Standard Package sideband requirements of UCIe Specification Revision 3.0, and which correctly conditions all PMO and SFES operational modes on the outcomes of the corresponding negotiation handshakes performed during link initialization.

1.4.3 Implement the Main State Machine

The third objective is to implement the Main State Machine, which integrates the Link Training State Machine and the RDI state machine into a unified control subsystem governing the complete operational lifecycle of the UCIe physical layer. The deliverable encompasses the LTSM controller, sequencing all eight top-level states and the thirteen sub-states within the mainband training phase, together with the RDI state machine, which mediates state transitions on the interface between the physical layer and the D2D Adapter. The combined subsystem must correctly coordinate state enable and completion handshakes between the two machines, enforce the specification-defined transition conditions including timeout detection and error escalation, and provide standardized control and status interfaces to the register file, sideband subsystem, and mainband datapath. The expected outcome is a verified state machine integration that drives the

physical layer through its full initialization, calibration, training, retraining, and error recovery sequences, including the shared calibration infrastructure used across multiple training sub-states, in strict conformance with UCIe Specification Revision 3.0.

1.4.4 Implement the Register File

The fourth objective is to implement the UCIe register file, which serves as the central configuration and status interface between software and the physical layer hardware subsystems. The deliverable encompasses both the UCIe Link DVSEC register block and the PHY-specific register space, covering capability advertisement, control, status monitoring, training configuration, lane mapping, and hardware-assisted error capture and logging functions. The register file must correctly enforce per-field access attributes as defined by the specification, capture hardware-driven status and error conditions from the LTSM and mainband datapath, and distribute software-programmed configuration values to the sideband subsystem, main state machine, and mainband datapath through well-defined internal interfaces. The expected outcome is a register file that provides a complete and spec-compliant software-visible state of the physical layer, enables software control of link training initiation and retraining, and accurately reflects post-training outcomes including negotiated link width, lane reversal status, width degradation events, and LTSM state history.

1.5 Project Scope

1.5.1 Inclusions

This project is scoped to the digital implementation of the UCIe Standard Package physical layer in its entirety. The analog front-end circuitry, including transmitter drivers, receiver samplers, phase-locked loops, and phase interpolators, is treated as abstracted hardware whose behavior is exposed to the digital subsystem through a set of well-defined digital control and status signal interfaces. All design decisions and behavioral requirements described in this thesis are grounded exclusively in UCIe Specification Revision 3.0.

The included implementation targets the Standard Package configuration, supporting both x16 and x8 mainband operating widths. The mainband datapath encompasses byte-to-lane mapping, scrambling, training pattern generation and comparison, lane reversal detection and application, and width degradation. The Link Training State Machine covers all eight top-level states and the thirteen sub-states of the mainband training phase, together with the associated timeout and error escalation logic. The RDI state machine provides the interface state abstraction layer between the physical layer and the D2D Adapter. The sideband subsystem operates on a permanently powered auxiliary domain and supports Sideband Feature Extensions and Sideband Performant Mode Operation, both negotiated during link initialization. The register file encompasses the UCIe Link DVSEC register block and the PHY-specific register space, providing software-accessible capability, control, status, and error logging functions. All included subsystems are integrated into a unified UCIe PHY architecture that provides the complete digital functionality required for Standard Package link initialization, training, operation, and management.

1.5.2 Exclusions

The analog front-end circuitry of the UCIe physical layer falls outside the scope of this project. Transmitter drivers, receiver samplers, clock distribution networks, clock generation and timing adjustment circuits, and phase interpolators are not designed or modeled at the circuit level. Their behavior is instead represented through abstracted digital control and status interfaces at which the digital subsystem drives configuration values and receives training results. This boundary is consistent with the digital-only scope established in the problem statement and reflects the standard practice of separately sourcing analog IP in industrial chiplet design flows.

Among the optional and non-normative features defined by UCIe Specification Revision 3.0, several are explicitly excluded from this implementation. Priority Sideband Packet Transfer is not supported; the specification notes that further details for this feature are deferred to future engineering change notices, making a complete and stable implementation impractical within the current revision. L2 Sideband Power Down is excluded as it requires a dedicated low-power sideband signaling infrastructure that extends significantly beyond the core link training and operational scope of this project. Tx Adjustment during Runtime Recalibration is omitted on the same basis, as it introduces a runtime analog compensation path whose digital coordination logic is separable from the primary training sequence. PHY Compliance Testing Support is excluded because compliance testing infrastructure constitutes a self-contained verification overlay rather than a functional component of the operational physical layer. Multi-module configuration is not supported; this implementation targets a single Standard Package module, and the multi-module PHY logic required to aggregate modules at the RDI presents a distinct and additive design problem. All vendor-defined register blocks and extensions are likewise excluded, as their content is by definition outside the normative requirements of the specification. These exclusions do not affect the implementation of the mandatory Standard Package PHY functionality required for compliant link initialization, training, operation, and management.

1.5.3 Analog Abstraction Boundary

The analog abstraction boundary defines the interface at which the digital implementation of this project terminates and the behavior of the underlying analog circuitry begins. On the physical side of this boundary lie the bump-level serial signals of the mainband and sideband interfaces, including the per-lane data, valid, clock, and track transmitters and receivers, together with the sideband clock and data pins. The electrical characteristics of these signals, and the circuits that drive and sample them, are outside the scope of this project and are assumed to be provided by a separately sourced analog front-end.

On the digital side of the boundary, the implementation interacts with the analog front-end through two categories of signals. The first comprises outward control signals through which the digital subsystem programs analog operating parameters associated with calibration, timing adjustment, voltage reference selection, clock management, and link speed configuration. These values are computed by the mainband training sub-states and applied to the analog front-end at the appropriate points in the training sequence. The second comprises inward status and result signals through which the analog front-end reports outcomes back to the digital subsystem,

including per-lane training results, pattern-comparison outcomes, calibration feedback signals, and clock-detection indicators. The boundary is defined such that the digital subsystem neither models nor depends on the analog circuit implementation, making the design independent of the specific analog IP technology with which it is integrated.

1.6 Contributions

This project makes the following contributions:

Complete digital UCIe Standard Package PHY architecture. A structurally complete, specification-grounded digital implementation of the UCIe Standard Package PHY developed in conformance with UCIe Specification Revision 3.0 and serving as the foundation for the subsystem-level contributions described below. To the authors' knowledge, no equivalent openly documented, spec-driven digital PHY for the UCIe Standard Package exists as a publicly available academic or community reference.

Full Revision 3.0-compliant Link Training State Machine. A complete implementation of the UCIe Revision 3.0 Link Training State Machine, covering all mandatory initialization, calibration, training, retraining, and error recovery procedures across all eight top-level states and thirteen mainband training sub-states.

Sideband subsystem with PMO and SFES support. A sideband protocol engine operating on a permanently powered auxiliary domain, with correct negotiation and runtime enforcement of both Sideband Feature Extensions and Sideband Performant Mode Operation.

Integrated RDI state machine. An RDI state machine coordinated as a peer of the LTSM, with explicit handshaking at every state transition boundary, providing a complete and correct interface abstraction layer between the physical layer and the D2D Adapter.

Complete UCIe PHY Register Architecture. A fully access-attribute-correct register file spanning both the UCIe Link DVSEC and PHY-specific register spaces, with hardware-driven status capture and software-controlled link management.

Explicit analog abstraction methodology. A clearly defined and documented analog abstraction boundary that partitions the digital control and status interface from the underlying analog front-end, enabling the digital design to be integrated independently of any specific analog IP implementation.

Open and academically documented reference architecture. A transparent, fully documented design that can serve as a reference baseline for future academic and engineering work in spec-compliant UCIe Standard Package PHY development.

1.7 Thesis Organization

The remainder of this thesis is organized as follows. Chapter 2 provides the specification background necessary to understand the design decisions made in subsequent chapters, covering the UCIe layered architecture, the physical layer functional decomposition, the Link Training State

Machine state and sub-state definitions, the sideband protocol and its feature extensions, the RDI state machine, and the register organization defined by UCIe Specification Revision 3.0. Chapter 3 presents the overall system architecture of the implemented PHY, describing the top-level block decomposition, subsystem hierarchy, clocking structure, inter-subsystem control and data interfaces, and the rationale behind the architectural partitioning. Chapter 4 details the mainband datapath design, covering the mapper and demapper, scrambling, lane reversal, and width degradation logic. Chapter 5 describes the sideband subsystem design, including the packet serializer and deserializer, the standard and performant mode transmission engines, and the Sideband Feature Extensions negotiation flow. Chapter 6 presents the Main State Machine design, covering the LTSM controller and all training state and sub-state implementations, the RDI state machine, and their integration and inter-machine handshaking. Chapter 7 describes the register file design, covering the UCIe Link DVSEC and PHY-specific register spaces, access attribute enforcement, and the hardware status capture interfaces. Chapter 8 presents the verification approach and simulation results, demonstrating correct behavioral operation of each subsystem and the integrated PHY across the key training and operational scenarios. Chapter 9 concludes the thesis with a summary of the completed work, a discussion of the design's limitations, and directions for future work.

2.0 UCIe Standard Package PHY: Specification Analysis

2.1 Standard Package Module Overview

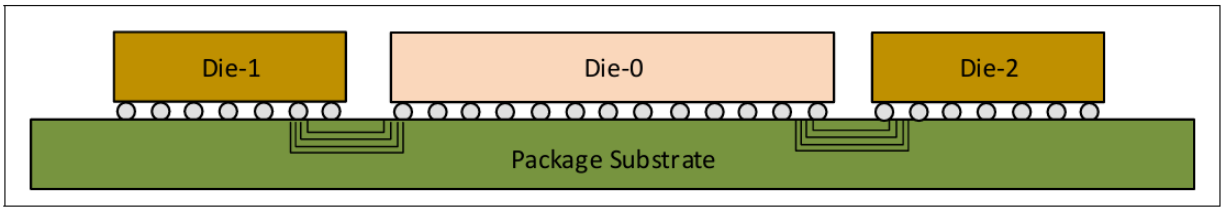


Figure 2-1: Standard Package interface

The UCIe physical link is organized around an entity called a Module, which represents the smallest atomic unit of structural implementation for the analog front end and the associated digital PHY logic. A Standard Package Module consists of two categories of physical connections: the mainband, which carries the primary data payload between dies, and the sideband, which carries the low-speed control channel used for parameter negotiation, register access, and link management coordination. Both categories are present on every module and operate independently in terms of power domain and clocking.

The Standard Package defines two native module widths. An x16 module provides sixteen data lanes per direction and constitutes the default and widest configuration supported by this implementation. An x8 operating configuration may arise either from a native x8 physical die or from width degradation applied to an x16 module that has encountered lane failures during link initialization. In either case, the sideband pin count remains unchanged: each module carries one data pin and one forwarded clock pin per direction regardless of mainband width. Unlike the Advanced Package, the Standard Package does not include redundant physical lanes for in-band repair; resilience against lane failures is instead addressed through width degradation, as covered in the relevant section of this chapter.

The Standard Package specification defines the physical interface between communicating dies but does not mandate a particular internal implementation architecture. In practical realizations, the PHY is commonly partitioned into digital logic responsible for protocol-related processing and analog circuitry responsible for clock generation, serialization, deserialization, and physical signal transmission. Consistent with the scope of this thesis, all analog functionality is abstracted through digital control and status interfaces, allowing the PHY behavior defined by the specification to be implemented and verified entirely within a synthesizable digital environment.

2.1.1 Module Signal Structure

The complete signal set of a Standard Package Module is defined in Table 5-26 of the specification. The signals fall into three functional groups: mainband data and associated timing signals, sideband signals, and power supply references. The mainband group is further subdivided by direction and by signal function within the transmit and receive paths.

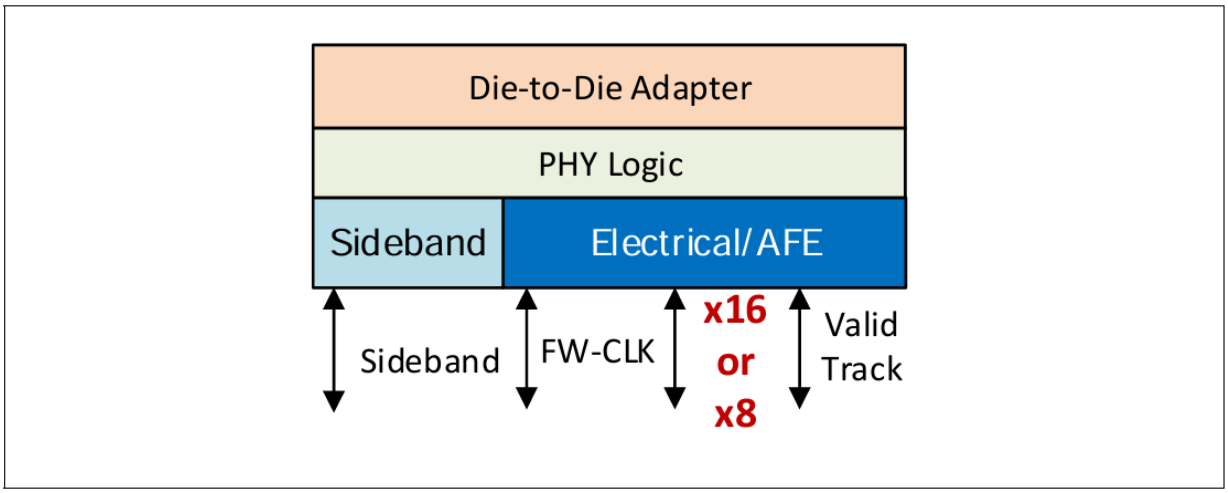


Figure 2-2: Single module configuration: Standard Package

Mainband Data Signals. On the transmit side, the module presents sixteen single-ended data pins designated TXDATA [15:0]. For an x8 operating configuration, only the lower eight members of this bus, TXDATA [7:0], are active; the transmitters of the remaining lanes are placed in a tri-state condition. The receive side presents a corresponding set of pins, RXDATA [15:0], with the same conventions for inactive lanes. Data symbols are transmitted concurrently across all active lanes at the negotiated mainband data rate.

Valid Signal. Each direction of the mainband includes one dedicated Valid signal: TXVLD on the transmit side and RXVLD on the receive side. The Valid signal serves a dual purpose. First, it gates the clock distribution network for all data lanes within the module, enabling rapid idle entry and exit without disturbing the forwarded clock itself. Second, it provides the Valid framing mechanism that delimits meaningful data intervals on the mainband. The transmitter implementation of the Valid signal is electrically equivalent to that of a regular data lane.

Forwarded Clock Signals. The mainband clock is forwarded from transmitter to receiver using two single-ended pins per direction, representing two phases of the same fundamental clock: TXCKP and TXCKN on the transmit side, and RXCKP and RXCKN on the receive side. Phase-1 and Phase-2 together form the timing reference used by the receiver to sample incoming data across all active lanes within the module. The forwarded clock serves as the timing reference for all active data lanes, and the relationship between its two phases and the incoming data eye is established and validated during the link training process, ensuring reliable sampling across the negotiated operating mode.

Track Signal. One Track pin is present per direction: TXTRK on the transmit side and RXTRK on the receive side. The Track signal allows the receiving PHY to compensate for slow-varying conditions

such as voltage and temperature drift by observing a periodic reference pattern transmitted by the remote PHY on this lane. When requested by the remote partner over the sideband, the transmitter drives a clock pattern aligned with Phase-1 of the forwarded clock on the Track lane. The Track lane is functionally separate from the data and clock lanes and is held low when not actively used.

Sideband Signals. Each module carries four sideband pins: TXDATASB, TXCKSB, RXDATASB, and RXCKSB. The transmit pair consists of a serial data output and its associated forwarded clock output; the receive pair provides the corresponding data input and clock input from the remote die. The sideband clock operates at a fixed frequency of 800 MHz, independent of the mainband data rate, and the sideband logic resides in an always-on power domain. These four pins constitute the entirety of the low-speed control channel for the module and are sufficient to carry all link training messages, register access transactions, and power management coordination required during the link lifetime.

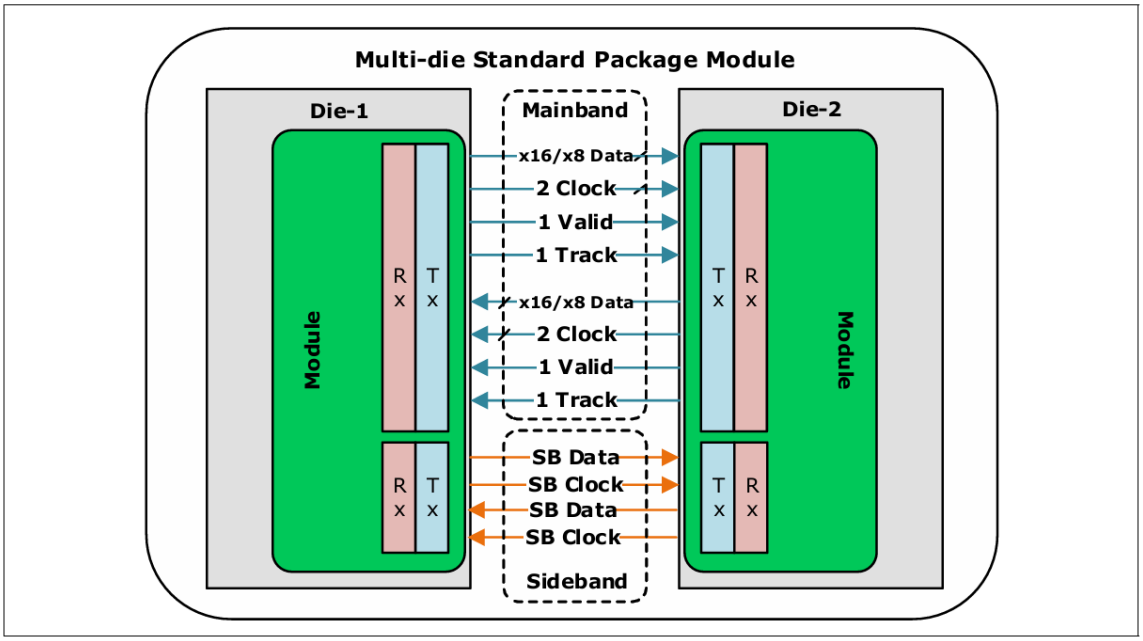


Figure 2-3: x16 or x8 Standard Package Module

Power Supply References. Three supply references accompany each module: VSS provides the ground reference, VCCIO supplies the I/O circuits of the mainband transmitters and receivers, and VCCAON provides the always-on auxiliary supply that powers the sideband logic and ensures the sideband channel remains operational even when the mainband power domain is reduced. The separation between VCCIO and VCCAON reflects the architectural requirement that the sideband remain reachable at all times, including during mainband low-power states.

2.1.2 x16 and x8 Width Configurations

The Standard Package supports two operating width configurations that determine the number of active data lanes, the byte-to-lane assignment, and the resulting aggregate bandwidth of the link. Both configurations share the same module signal structure and bump map layout defined in the preceding sections; the distinction between them lies in which lanes carry valid data and which are placed in an electrically inactive condition.

x16 Full-Width Operation. In the x16 configuration, all sixteen data lanes of the module are active in each direction. Data is transmitted one byte per lane, with Byte 0 assigned to Lane 0, Byte 1 to Lane 1, and so on through Byte 15 on Lane 15. Within each byte, bit zero is transmitted first, followed by the remaining bits in sequence across eight consecutive unit intervals. All sixteen transmitters are active, all sixteen receivers are enabled, and the forwarded clock serves as the timing reference for the full lane population. This configuration delivers the maximum data throughput available to a single Standard Package module.

x8 Operating Configuration. The x8 configuration reduces the active lane count to eight per direction. Two distinct x8 lane assignments are defined by the specification. The lower-half assignment activates Lanes 0 through 7, while the upper-half assignment activates Lanes 8 through 15. The choice between lower and upper halves is governed by lane health: the half that is free of detected faults is selected, and the half containing faulty lanes is disabled. When a lane is disabled, its transmitter is placed in a tri-state condition and its receiver is turned off. The forwarded clock, valid, and track signals remain associated with the active half and operate without modification. For a native x8 module, the same two half-selections apply, with a further degradation to x4 available using either Lanes 0 through 3 or Lanes 4 through 7 from within the active half.

Byte Mapping Under Width Reduction. In the x8 lower-half configuration, Bytes 0 through 7 are assigned to the active Lanes 0 through 7, preserving the same one-byte-per-lane rule that applies in x16 operation. In the x8 upper-half configuration, Bytes 0 through 7 are remapped to the active Lanes 8 through 15, maintaining sequential assignment within the active lane set. This remapping ensures that the upper protocol layers observe a contiguous eight-byte interface regardless of which physical half of the module is carrying traffic.

Width Negotiation. The supported operating width of each die is communicated to the link partner during the initialization process through parameter exchange messages. The final link width is determined from the capabilities and lane availability reported by both link partners. The resolved width is subsequently used throughout training and normal operation. The mechanisms used to evaluate lane health and perform width degradation are discussed in the context of mainband initialization and training in later sections of this chapter.

2.1.3 Analog Abstraction Boundary

The UCIe specification defines the physical interface of a Standard Package Module in terms of external behavior: the signals present at the bump level, the electrical characteristics those signals must satisfy, and the timing and training procedures required to bring the link into operation. The specification does not mandate a specific internal partitioning of the PHY. It is the responsibility of

each implementation to allocate the required functions between digital logic and analog circuitry in a manner consistent with the target process technology and design constraints.

In practice, the functions required by the PHY divide naturally into two groups along an analog-digital boundary. On the analog side reside the functions that operate directly on continuous-time or high-frequency physical signals: clock generation through a phase-locked loop, clock distribution and gating, frequency division to produce the parallel-domain reference clock, high-speed serialization of outbound data, deserialization of inbound data sampled on the forwarded receive clock, and the tri-state output drivers that present or suppress signal activity on the physical bumps. These functions are tightly coupled to the electrical characteristics of the bump interface and are sensitive to process, voltage, and temperature variations in ways that are not amenable to purely digital modeling. On the digital side reside all protocol-level functions: byte-to-lane mapping and scrambling, linear feedback shift register pattern generation for training, pattern comparison for per-lane error detection, lane reversal, width degradation logic, valid framing, clock pattern generation for training purposes, clock activity detection on received clock signals, link training state machine operation, sideband protocol processing, and register file management. These functions are independent of the continuous-time physical medium and are fully expressible as synchronous logic operating on a parallel-domain clock derived from the analog clock generation hierarchy.

The boundary between these two domains is characterized by a set of interface signals that carry information in both directions. In the transmit direction, the digital domain delivers pre-serialization parallel data words — one word per lane per parallel clock cycle — along with control signals that govern the operating mode of each signal group. In the receive direction, the analog domain returns recovered parallel data words assembled from the serially received stream, together with framing status derived from the received valid signal. Clocks generated by the analog domain are distributed to the digital domain as synchronous references for all protocol-level processing. Additional status and timing information may be provided across the abstraction boundary as required to support training, calibration, and link-management procedures defined by the specification.

For the sideband channel, the same structural separation applies. The analog front end of the sideband performs serial-to-parallel conversion on the received serial stream and parallel-to-serial conversion on the outbound stream, driven by a fixed 800 MHz serialization clock that governs the serial-domain bit rate only; all digital sideband protocol logic operates at 100 MHz, derived from the same 800 MHz source through a divide-by-eight stage internal to the sideband subsystem. The digital sideband logic operates on the resulting parallel bus and has no direct connection to the physical sideband bump pins.

Consistent with the scope of this thesis, all analog functionality described above is treated as an abstracted entity presenting well-defined digital control and status interfaces to the surrounding logic. The digital PHY implementation targets this abstraction boundary as its outer interface, such that the complete design can be described, simulated, and verified within a synthesizable digital environment without requiring process-specific analog models. The detailed architecture of the

digital domain, and the precise definition of its interface with the analog front end, are presented in Chapter 3.

2.1.4 PHY Layer Position within the UCIe Stack

The Physical Layer occupies the lowest position in this hierarchy and is responsible for all functions directly associated with the physical link: link training and initialization, lane reversal, scrambling and descrambling, sideband initialization and packet transfer, clock forwarding, and the analog front end that drives and receives physical signals at the bump level. Above the Physical Layer sits the Die-to-Die Adapter, which handles reliable data transfer through CRC and retry mechanisms where required, arbitration and multiplexing across protocol stacks, link state management, and parameter negotiation with the remote link partner. Above the D2D Adapter sits the Protocol Layer, which carries protocol-specific Flits in formats defined for PCIe, CXL, or Streaming protocols. Figure 2.x illustrates the relationship between the Protocol Layer, D2D Adapter, and Physical Layer within the UCIe communication stack..

The boundary between the Physical Layer and the D2D Adapter is defined by the Raw Die-to-Die Interface, commonly referred to as the RDI. The RDI abstracts the state and data path of the Physical Layer into a form that the Adapter can interact with independently of the details of link training, lane configuration, and physical signaling. On the data path, the RDI carries payload data between the Adapter and the Physical Layer using a parallel interface whose width scales with the negotiated link configuration. On the control path, the RDI exposes a state machine that the Adapter uses to request link state transitions — including transitions to Active, Retrain, L1, L2, LinkReset, and Disabled states — and that the Physical Layer uses to report the current link status back to the Adapter. The separation provided by the RDI ensures that the Adapter and Protocol Layer are insulated from the details of physical link management and can operate identically regardless of the underlying package type, data rate, or lane configuration.

The boundary between the D2D Adapter and the Protocol Layer is defined by the Flit-aware Die-to-Die Interface, referred to as the FDI. The FDI carries fully formatted protocol Flits in the negotiated Flit format and exposes a higher-level link state machine to the Protocol Layer. While the FDI is part of the complete UCIe stack, it lies above the scope of this thesis, which is confined to the Physical Layer and its RDI boundary.

The sideband channel plays a role that spans all three layers. At the physical level, the Physical Layer is responsible for framing, serializing, and transporting sideband packets across the link. The D2D Adapter and Protocol Layer originate and consume sideband messages, forwarding them to the Physical Layer over the RDI for transmission to the remote die. The Physical Layer routes received sideband packets upward to the appropriate destination based on the source and destination identifiers carried in each packet header. This arrangement means that the sideband channel, while physically managed by the PHY, serves as a shared communication medium for all layers of the stack.

Within the Physical Layer itself, the specification identifies two internal sub-components relevant to this implementation. Within the Physical Layer, the specification distinguishes between PHY

Logic and the Analog Front End. Consistent with the abstraction methodology established in Section 2.1.3, this thesis focuses on the PHY Logic portion of the design while treating the Analog Front End as an external implementation-dependent component.

2.2 Mainband Specification

The mainband constitutes the primary data path of the UCIe link, carrying protocol Flits between dies at the negotiated operating data rate. This section addresses the specification requirements governing the logical organization and timing of mainband data transfer, the mechanisms used to manage idle and active intervals on the physical lanes, the scrambling and lane reversal procedures required for signal integrity and physical interoperability, and the width degradation mechanism applicable to Standard Package modules. The sideband channel, which operates independently of the mainband, is addressed separately in Section 2.3.

2.2.1 Byte-to-Lane Mapping and Valid Framing

Byte-to-Lane Mapping. Data packets are organized as sequences of bytes and distributed across the active physical lanes according to a fixed sequential assignment. Each byte is carried on a dedicated lane, with Byte 0 assigned to Lane 0, Byte 1 to Lane 1, and so on through the full active lane population. Within each byte, bit zero is transmitted first, followed by the remaining seven bits across eight consecutive unit intervals. This one-byte-per-lane assignment is preserved without modification regardless of the link operating width; a reduction in the active lane count does not alter the relationship between byte index and lane index within the active set.

Lane / UI	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0 - 7	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15
8 - 15	B16	B17	B18	B19	B20	B21	B22	B23	B24	B25	B26	B27	B28	B29	B30	B31
16 - 23	B32	B33	B34	B35	B36	B37	B38	B39	B40	B41	B42	B43	B44	B45	B46	B47
24 - 31	B48	B49	B50	B51	B52	B53	B54	B55	B56	B57	B58	B59	B60	B61	B62	B63
32 - 39	B64	B65	B66	B67	B68	B69	B70	B71	B72	B73	B74	B75	B76	B77	B78	B79
40 - 47	B80	B81	B82	B83	B84	B85	B86	B87	B88	B89	B90	B91	B92	B93	B94	B95
48 - 55	B96	B97	B98	B99	B100	B101	B102	B103	B104	B105	B106	B107	B108	B109	B110	B111
56 - 63	B112	B113	B114	B115	B116	B117	B118	B119	B120	B121	B122	B123	B124	B125	B126	B127
64 - 71	B128	B129	B130	B131	B132	B133	B134	B135	B136	B137	B138	B139	B140	B141	B142	B143
72 - 79	B144	B145	B146	B147	B148	B149	B150	B151	B152	B153	B154	B155	B156	B157	B158	B159
80 - 87	B160	B161	B162	B163	B164	B165	B166	B167	B168	B169	B170	B171	B172	B173	B174	B175
88 - 95	B176	B177	B178	B179	B180	B181	B182	B183	B184	B185	B186	B187	B188	B189	B190	B191
96 - 103	B192	B193	B194	B195	B196	B197	B198	B199	B200	B201	B202	B203	B204	B205	B206	B207
104 - 111	B208	B209	B210	B211	B212	B213	B214	B215	B216	B217	B218	B219	B220	B221	B222	B223
112 - 119	B224	B225	B226	B227	B228	B229	B230	B231	B232	B233	B234	B235	B236	B237	B238	B239
120 - 127	B240	B241	B242	B243	B244	B245	B246	B247	B248	B249	B250	B251	B252	B253	B254	B255

Figure 2-4: Byte map for x16 interface

In the full-width x16 configuration, Bytes 0 through 15 are assigned to Lanes 0 through 15, with all sixteen transmitters and all sixteen receivers active. This configuration delivers the maximum data throughput available to a single Standard Package module.

In a width-reduced x8 lower-half configuration, Bytes 0 through 7 are assigned to the active Lanes 0 through 7. In the x8 upper-half configuration, Bytes 0 through 7 are remapped to the physically active Lanes 8 through 15, preserving sequential byte-to-lane assignment within the active set. In both cases, the eight-byte protocol interface presented to the layers above remains contiguous and independent of which physical half of the module is carrying traffic.

For a Standard Package module operating in a native x8 configuration, or one that has undergone x16-to-x8 degradation with the x8 operating mode established, a further width reduction to x4 is permitted when lane failures are contained within a single contiguous quarter of the active lane set. In the x4 lower-quarter configuration, Bytes 0 through 3 are assigned to Lanes 0 through 3. In the x4 upper-quarter configuration, Bytes 0 through 3 are remapped to the active Lanes 4 through 7, maintaining the sequential assignment rule within the active set. The four-byte protocol interface presented upstream is contiguous in both cases. The x4 operating mode is gated by the Standard Package Module Width capability bit or by the negotiated x8 parameter exchange during initialization; neither x4 encoding is applicable to a module that has not established x8 operation through one of these paths.

Valid Framing. The Valid signal governs the temporal boundaries of meaningful data on the mainband. For each 8-UI data transfer, the transmitter asserts Valid for the first four unit intervals and de-asserts it for the remaining four. This 4-high, 4-low pattern within each 8-UI block constitutes the valid framing envelope, allowing the receiver to distinguish active transfer intervals from idle periods and to establish byte boundaries across back-to-back transfers. Following the first valid data transfer after entering the Active state, each subsequent transfer must begin at an integer multiple of 8 UI from the rising edge of the Valid signal of that first transfer; this 8-UI block alignment is enforced by the transmitter and tracked by the receiver throughout the Active state. At 48 GT/s and 64 GT/s, additional protection is applied through a Hamming distance-4 encoding on the Valid signal, providing double-error detection and single-error correction to reduce the frequency of retraining events caused by Valid framing errors.

Clock Gating and Idle State. When no data transfer is active and Valid is held low, the forwarded mainband clock must be gated after a postamble period whose duration depends on the clocking mode. Under half-rate clocking, the postamble is sixteen unit intervals; under quarter-rate clocking, it is thirty-two unit intervals. Once the postamble requirement is satisfied, the parked clock level must alternate between differential-high and differential-low across consecutive clock-gating events in strobe mode, while the data lanes hold a defined idle state until the next transmission.

In a width-degraded x8 lower-half configuration, Bytes 0 through 7 are assigned to Lanes 0 through 7. In the x8 upper-half configuration, Bytes 0 through 7 are remapped to the physically active Lanes 8 through 15, preserving sequential assignment within the active set while presenting a contiguous eight-byte interface to the layers above. This remapping ensures that the protocol-layer view of the data is independent of which physical half of the module is carrying traffic.

Lane	8	9	10	11	12	13	14	15
	or							
UI \ Lane	0	1	2	3	4	5	6	7
0 - 7	B0	B1	B2	B3	B4	B5	B6	B7
8 - 15	B8	B9	B10	B11	B12	B13	B14	B15
16 - 23	B16	B17	B18	B19	B20	B21	B22	B23
24 - 31	B24	B25	B26	B27	B28	B29	B30	B31
32 - 39	B32	B33	B34	B35	B36	B37	B38	B39
40 - 47	B40	B41	B42	B43	B44	B45	B46	B47
48 - 55	B48	B49	B50	B51	B52	B53	B54	B55
56 - 63	B56	B57	B58	B59	B60	B61	B62	B63
64 - 71	B64	B65	B66	B67	B68	B69	B70	B71
72 - 79	B72	B73	B74	B75	B76	B77	B78	B79
80 - 87	B80	B81	B82	B83	B84	B85	B86	B87
88 - 95	B88	B89	B90	B91	B92	B93	B94	B95
96 - 103	B96	B97	B98	B99	B100	B101	B102	B103
104 - 111	B104	B105	B106	B107	B108	B109	B110	B111
112 - 119	B112	B113	B114	B115	B116	B117	B118	B119
120 - 127	B120	B121	B122	B123	B124	B125	B126	B127
128-135	B128	B129	B130	B131	B132	B133	B134	B135
136-143	B136	B137	B138	B139	B140	B141	B142	B143
144-151	B144	B145	B146	B147	B148	B149	B150	B151
152-159	B152	B153	B154	B155	B156	B157	B158	B159
160-167	B160	B161	B162	B163	B164	B165	B166	B167
168-175	B168	B169	B170	B171	B172	B173	B174	B175
176-183	B176	B177	B178	B179	B180	B181	B182	B183
184-191	B184	B185	B186	B187	B188	B189	B190	B191
192-199	B192	B193	B194	B195	B196	B197	B198	B199
200-207	B200	B201	B202	B203	B204	B205	B206	B207
208-215	B208	B209	B210	B211	B212	B213	B214	B215
216-223	B216	B217	B218	B219	B220	B221	B222	B223
224-231	B224	B225	B226	B227	B228	B229	B230	B231
232-239	B232	B233	B234	B235	B236	B237	B238	B239
240-247	B240	B241	B242	B243	B244	B245	B246	B247
248-255	B248	B249	B250	B251	B252	B253	B254	B255

Figure 2-5: Byte to Lane mapping for Standard package x16 degraded to x8

Valid Framing. The Valid signal governs the temporal boundaries of meaningful data on the mainband. For each 8-UI data transfer, the transmitter asserts the Valid signal for the first four-unit intervals and de-asserts it for the remaining four. This 4-high, 4-low pattern within each 8-UI block constitutes the Valid framing envelope. The Valid framing pattern identifies the boundaries of meaningful data transfers and allows the receiver to distinguish active transfer intervals from idle periods. At 48 GT/s and 64 GT/s, additional protection is applied to the Valid framing mechanism through a Hamming distance-4 encoding, providing double-error detection and single-error correction capability on the Valid signal to minimize retraining events caused by Valid framing errors.

Following the first valid data transfer after entering the Active state, each subsequent transfer must begin at an integer multiple of 8 UI from the rising edge of the Valid signal of that first transfer. This 8-UI block alignment is enforced by the transmitter and tracked by the receiver throughout the Active state, ensuring that byte boundaries remain coherent across back-to-back and non-contiguous transfers within the same session. The Valid framing pattern identifies the boundaries of meaningful data transfers and allows the receiver to distinguish active transfer intervals from idle periods.

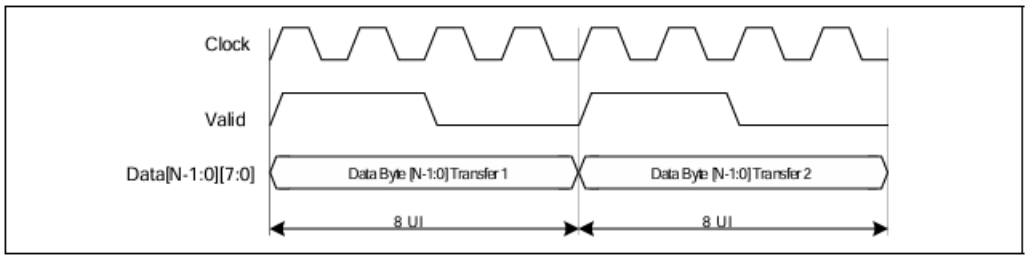


Figure 2-6: Valid framing example

Clock Gating and Idle State. When no data transfer is active and the Valid signal is held low, the forwarded mainband clock must be gated after a postamble period whose duration depends on the clocking mode. Under half-rate clocking, the postamble is sixteen-unit intervals; under quarter-rate clocking, it is thirty-two unit intervals. Once the postamble requirement is satisfied and the clock is gated, the parked clock level must alternate between differential-high and differential-low across consecutive clock-gating events in strobe mode, while the data lanes hold a defined idle state until the next transmission. For terminated links, data lane transmitters hold the last transmitted bit for the duration of the hold period before transitioning to a high-impedance condition; for unterminated links, some transmitters are permitted to continue toggling at normal transition density without advancing the scrambler state. In all cases, the total idle period must be an integer multiple of 8 UI to preserve alignment with the Valid framing grid.

At data rates of 48 GT/s and 64 GT/s, the forwarded clock lanes must operate in continuous clock mode regardless of the Valid state. In this mode, the clock continues to toggle freely during idle intervals, eliminating the postamble requirement and simplifying receiver timing recovery at the highest supported data rates.

Free-running clock mode allows the forwarded clock to remain continuously active during idle periods and may be selected through capability negotiation during initialization. Support for this mode is mandatory and enables dynamic clock gating to be disabled for normal operation or debug purposes.

2.2.2 Scrambling and Training Pattern Generation (LFSR)

The UCIe specification employs a common Linear Feedback Shift Register (LFSR) structure for both mainband data scrambling and training-pattern generation. Using a single sequence definition ensures consistent transmitter and receiver behavior across normal operation, link initialization, and diagnostic procedures. The use of a common LFSR definition for both purposes ensures that the transmitter and receiver employ the same sequence at every stage, which is a prerequisite for coherent comparison.

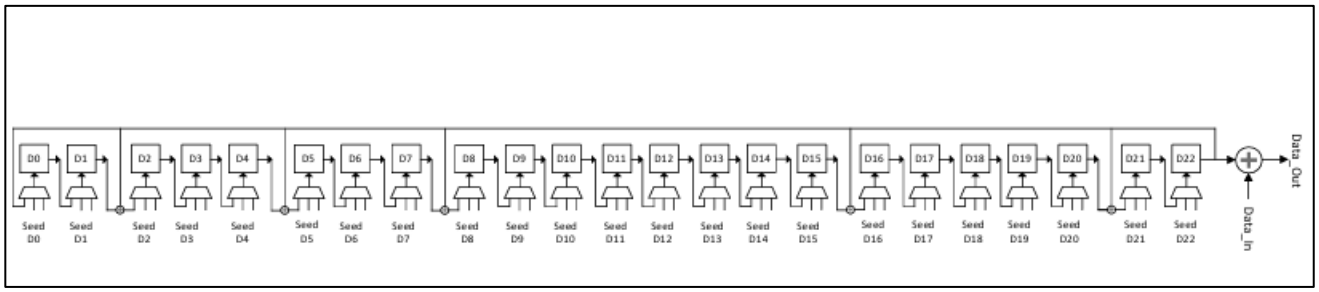


Figure 2-7: LFSR implementation

LFSR Polynomial and Seed Assignment. The LFSR is defined by the polynomial $G(X) = X^{23} + X^{21} + X^{16} + X^{8} + X^{5} + X^{2} + 1$, which is identical to the polynomial used by the PCIe specification. The register is twenty-three bits wide. The initial state of the LFSR is determined by a seed value that is assigned on a per-lane basis according to the logical lane number, using a modulo-8 mapping. Eight distinct seed values are therefore sufficient to initialize all lanes in any supported width configuration: lanes 0 through 7 are each assigned a unique seed, and lanes 8 through 15 reuse the seed set of lanes 0 through 7 by taking the lane number modulo 8. The specification permits each transmitter to implement either a dedicated per-lane LFSR instance or a single shared LFSR with differentiated tap points for multiple lanes; both approaches are stated to be equivalent in terms of the generated sequence.

Scrambling Mode. During data transfer in the Active state, the LFSR operates in scrambling mode. On the transmit side, each data lane XORs the outgoing data stream with the PRBS output of its associated LFSR, producing a scrambled stream for transmission over the physical lane.

On the receive side, each data lane XORs the recovered parallel data with the output of a locally running LFSR initialized to the same seed, recovering the original data. Correct descrambling requires that the transmit and receive LFSR instances begin from the same seed state and advance in lockstep throughout the data transfer interval. Scrambling is mandatory during mission-mode data transfer and may not be disabled for normal link operation.

Table 2-1: Per-lane LFSR seed values

Logical Lane	Seed Value
0	23'h1DBFBC
1	23'h0607BB
2	23'h1EC760
3	23'h18C0DB
4	23'h010F12
5	23'h19CFC9
6	23'h0277CE
7	23'h1BB807

Pattern Generation Mode. During link training, the LFSR operates in pattern generation mode, in which the transmitter suppresses the incoming data input and outputs the raw PRBS sequence directly onto the physical lane. The transmitted PRBS pattern, accompanied by the forwarded clock and Valid signal, is received by the remote die and compared against a locally generated reference pattern produced by an LFSR instance initialized to the same lane-specific seed. Both pattern generators must start and advance in synchrony, such that the comparison at each unit interval reflects whether the physical transmission was error-free. Any mismatch between the received pattern and the locally predicted sequence at a given UI is recorded as an error event.

Per-Lane Identification Pattern. In addition to the PRBS pattern, the specification defines a fixed per-lane identification pattern used during specific training substates to establish lane correspondence between the transmitter and receiver. Each lane transmits a unique fixed word constructed from its logical lane number embedded in a recognizable framing structure. This pattern allows the receiver to determine which physical receive lane corresponds to each physical transmit lane without relying on advance knowledge of any lane reversal or remapping that may be in effect.

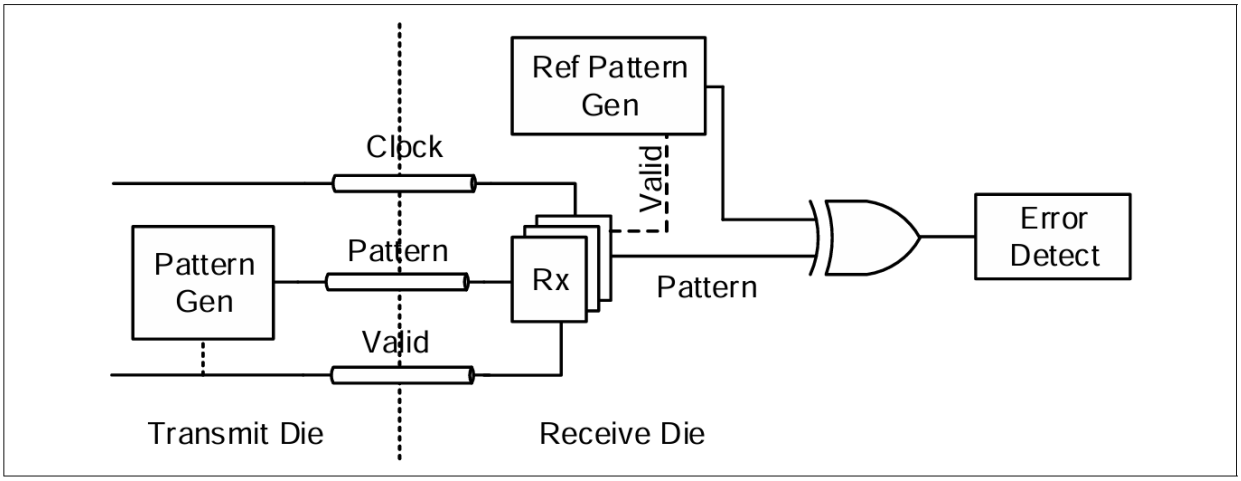


Figure 2-8: Test and training logic

Comparison Infrastructure. The receive die must support two comparison operating modes. In per-lane comparison mode, the error detection logic maintains an independent error indicator for each active lane. If the number of comparison mismatches on a given lane exceeds a configurable threshold during the comparison window, the error indicator for that lane is latched and held for the remainder of the test interval. The per-lane results constitute the primary mechanism for identifying which individual lanes have failed or degraded and are accessible to the local PHY through sideband reads.

In aggregate comparison mode, mismatches detected at any UI on any lane within the module are accumulated into a sixteen-bit error counter, with lane errors combined by logical OR before counting. This aggregate mode is used for module-level margin characterization and raw bit error rate estimation rather than for individual lane fault isolation.

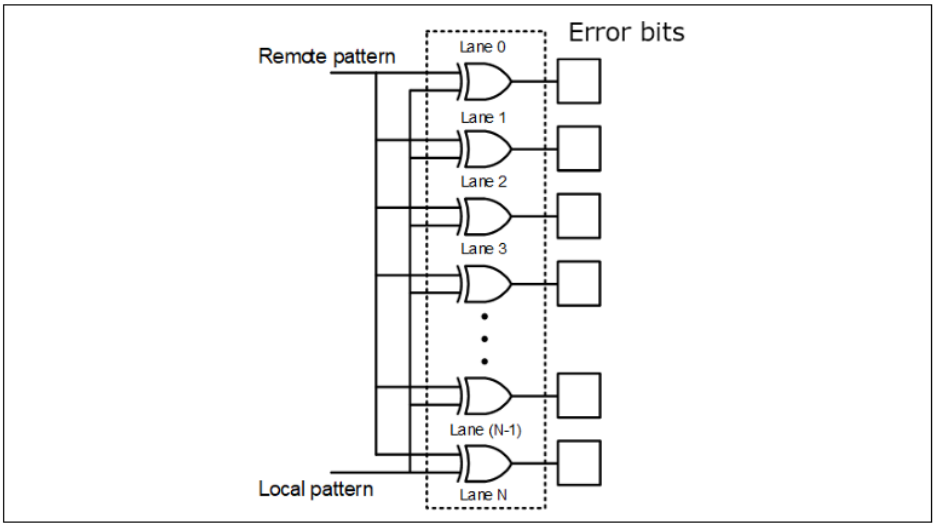


Figure 2-9: Lane failure detection

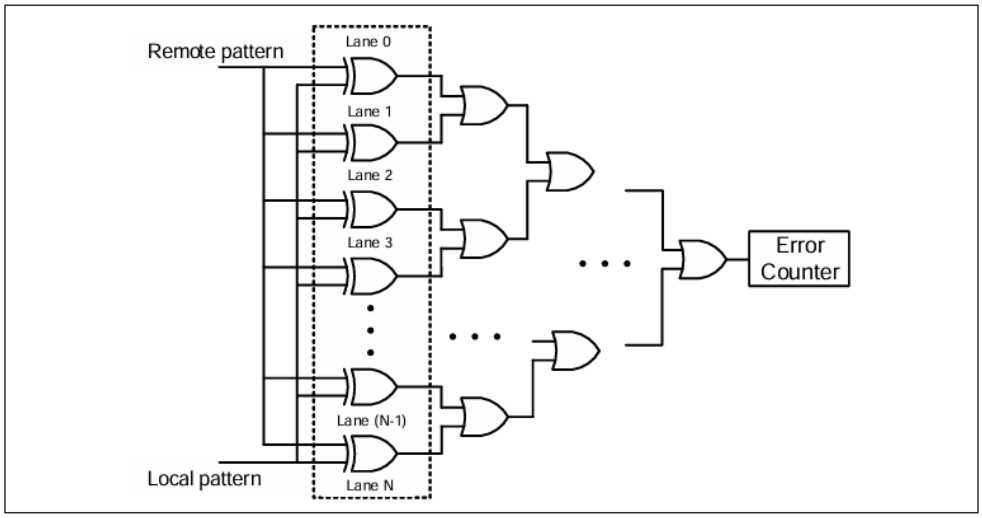


Figure 2-10: All Lane error detection

2.2.3 Lane Reversal

Purpose and Physical Motivation. Lane reversal is a mechanism that accommodates die assembly configurations in which the physical data lane connections between two dies are ordered in reverse. This situation arises when one die is oriented such that its transmit lane 0 bump connects to the receive lane N−1 bump of the remote die, lane 1 to lane N−2, and so on through the full active lane set. Without a correction mechanism, such a reversed connection would prevent meaningful data transfer by misaligning all lane-to-byte assignments. The UCIe specification mandates that all compliant devices support lane reversal detection and correction to ensure that this physical orientation does not restrict package assembly choices.

Scope of Reversal. Lane reversal applies exclusively to data lanes. The Track, Valid, forwarded clock, and sideband signals are symmetric with respect to the lane axis and must not be reversed. Redundant lanes in Advanced Package implementations are also subject to reversal when it applies, but these are absent in the Standard Package and are therefore not relevant to this implementation. Importantly, the specification mandates that lane reversal correction be applied at the transmitter only. The transmitter reorders the logical lane assignments on its outputs to compensate for the physical reversal, while the receiver requires no modification to its lane mapping.

Lane ID Assignment. To enable reversal detection, each logical data lane within a module is assigned a unique Lane ID. For a Standard Package module, the sixteen logical data lanes carry Lane IDs 0 through 15. These IDs are conveyed to the remote die using a fixed sixteen-bit Per-Lane ID pattern transmitted during a dedicated training substate. The pattern format consists of a four-bit preamble of 0101, an eight-bit Lane ID field in which bit 0 of the Lane ID occupies bit 4 of the pattern, followed by a four-bit postamble of 1010. This structure embeds the lane identifier within a recognizable framing sequence that the receiver can detect with confidence. The Per-Lane ID

pattern is transmitted without scrambling, and detection on a given receive lane is considered successful if at least sixteen consecutive correct iterations of the pattern are observed.

Discovery Procedure Lane reversal detection is performed during the MBINIT.REVERSALMB training substate. During this procedure, each die transmits the Per-Lane ID pattern across the active data lanes while the remote receiver determines the correspondence between received patterns and expected logical lane identifiers. Based on the comparison results exchanged between the link partners, the transmitter either retains the original lane ordering or applies lane reversal before training proceeds. If neither orientation produces a valid lane mapping, the link transitions to the training error state. The detailed handshake sequence, pattern iteration count, and sideband message exchange governing this substate are described in Section 2.4.

The results are interpreted according to a majority rule. If the majority of active data lanes report successful detection — meaning the received Per-Lane ID patterns matched the locally expected patterns — lane reversal is not required and the procedure advances to the next training substate. If exactly half of the lanes report success, the specification treats this as a reversal condition and lane reversal is applied. If fewer than half report success, the local die applies lane reversal on its transmitters and repeats the pattern transmission and comparison sequence. If the repeated attempt produces a majority of successful lanes, lane reversal is confirmed and retained for the remainder of the link's operational lifetime. If neither attempt yields a majority of successful detections, the link exits to the training error state. Following a successful outcome in either direction, a completion handshake is exchanged before the LTSM advances to the repair substate.

Interaction with Width Degradation. When lane reversal is combined with width degradation, the reversal is applied within the active lane set after the degraded lane map has been determined. The Per-Lane ID pattern is transmitted only on the active lanes corresponding to the selected half of the module, and the majority rule is evaluated across that reduced lane population. This ensures that reversal detection remains coherent regardless of which half of the module is carrying traffic.

2.2.4 Width Degradation

Principle. The Standard Package does not provide redundant physical lanes for in-band repair. When one or more data lanes are found to be non-functional during link initialization or training, the resilience mechanism available to a Standard Package module is width degradation: a reconfiguration of the link to a narrower active lane set that excludes the faulty lanes entirely. This approach trades available bandwidth for continued link operability and is the only lane-level fault-recovery mechanism defined for Standard Package implementations.

Degradation Levels and Feasibility Conditions. For an x16 Standard Package module, degradation is available to an x8 operating configuration. The specification permits degradation to x8 only if all lanes encountering errors are contained within a single contiguous half of the module: either Lanes 0 through 7 or Lanes 8 through 15. The error-free half is then selected as the active lane set. If errors span both halves such that neither half is entirely free of faults, width degradation is not feasible and the link cannot be recovered by this mechanism.

For a Standard Package module operating in x8 mode — either as a native x8 implementation or following x16-to-x8 degradation — a further degradation to x4 is available under the same containment condition. All lane failures must be contained within either Lanes 0 through 3 or Lanes 4 through 7, and the fault-free quarter is selected as the active lane set. The x4 degradation encodings are only applicable when the Standard Package Module Width capability bit has been set, or when the UCIe-S x8 parameter was exchanged as set during the mainband initialization parameter exchange.

Lane Map Encoding. The specification defines a three-bit encoding to represent all supported lane configurations for Standard Package modules. This encoding is carried in the sideband messages exchanged during degradation negotiation, allowing each die to communicate its locally determined functional lane set to its link partner.

Table 2-2: Standard Package logical lane map encodings

Encoding	Active Lane Set	Effective Width
011b	Lanes 0–15	x16
001b	Lanes 0–7	x8 lower half
010b	Lanes 8–15	x8 upper half
100b	Lanes 0–3	x4 lower quarter
101b	Lanes 4–7	x4 upper quarter
000b	None	Degradation not feasible

Physical Effect of Degradation. When a width degradation is applied, the transmitters of all lanes outside the selected active set are placed in a tri-state condition, and the corresponding receivers are disabled. The forwarded clock, Valid, and Track signals are unaffected by lane degradation and continue to operate on their dedicated lanes regardless of the active data lane configuration. Consequently, only the active data lane population changes; the timing and control infrastructure of the module remains unchanged. All byte-to-lane mapping within the active set follows the remapping rules described in Section 2.1.3.

Degradation During Initialization and Retraining. Width degradation is performed in two distinct contexts. During mainband initialization, the MBINIT.REPAIRMB substate provides an initial lane assessment and negotiation before high-speed training begins, allowing an x8 or x4 configuration to be established prior to any data-to-clock calibration. During the mainband training sequence, the MBTRAIN.REPAIR substate provides a second degradation opportunity if training reveals additional lane failures that were not detected during initialization. In both cases, the negotiation is performed through a sideband message exchange in which each die communicates its locally determined lane map encoding to its partner, and the receiver of each die is configured to match

the partner's reported active lane set. The detailed message sequence for both substates is described in Section 2.4.

Constraint on Successive Degradation. Only one degradation step is permitted per training session. If a lane fault is detected after a degradation has already been applied during the current session, no further reduction in link width is available, and the link exits to the training error state. A fresh degradation opportunity is available only after the link has completed a full retraining cycle from the appropriate re-entry point.

If degradation is not feasible — because no contiguous half or quarter of the module is entirely free of faults, or because the lane map encoding resolves to the none encoding — the link transitions to the training error state and cannot be brought to an operational condition until a subsequent link initialization or retraining sequence is initiated.

2.2.5 Training Pattern Types

The UCIe specification defines a set of pattern types that are transmitted across the mainband during link initialization and training. These patterns serve distinct purposes at different stages of the training sequence and are applied selectively to the data lanes, the Valid lane, and the forwarded clock lanes according to the requirements of each training substate.

PRBS Pattern. The pseudo-random bit sequence pattern, generated by the LFSR defined in Section 4.4.1, is the primary pattern type used on the active data lanes. The transmitter outputs the raw LFSR sequence without scrambling, and the receiver compares the incoming stream against a locally generated reference sequence initialized to the same lane-specific seed. The PRBS pattern is employed throughout multiple training substates requiring dynamic data transitions for timing, voltage, and link-quality evaluation. The specific substates in which it is applied are identified in Section 2.4.

Per-Lane ID Pattern. The Per-Lane ID pattern is a fixed sixteen-bit lane-specific sequence used exclusively during the lane reversal detection substate of mainband initialization. Its format and purpose are described in Section 2.2.3. Unlike the PRBS pattern, the Per-Lane ID pattern is transmitted without scrambling and is evaluated through dedicated detection logic at the receiver.

Fixed All-Zeros Pattern. A third data pattern option drives all active data lane transmitters to a constant low level. This pattern is used in specific calibration contexts where a known static state on the data lanes is required, without the toggling activity of the PRBS sequence.

Valid Training Pattern. The Valid lane carries a dedicated training pattern during Valid lane calibration substates. This pattern has a defined alternating structure that allows the receiver to evaluate the alignment between the transmitted Valid signal and the forwarded receive clock. The Valid training pattern is distinct from the operational Valid framing signal described in Section 2.2.1 and is used exclusively during training. In substates where the Valid training pattern is not required, the Valid lane transmitter is held low.

Clock Pattern Modes. The forwarded clock lanes support two transmission modes during training. In the first mode, the clock is driven with the operational differential toggling waveform representing the Phase-1 and Phase-2 clock relationship, providing an active timing reference for receiver calibration. In the second mode, the clock transmitter is held at a static differential-low condition, suppressing clock activity on the physical bumps. The selected clock transmission mode depends on the requirements of the active training substate.

2.3 Sideband Specification

The sideband channel provides the low-speed serial control path between two communicating dies and operates independently of the mainband in terms of clocking, power domain, and protocol. Its primary function is to carry the structured packet traffic required for link initialization, parameter negotiation, register access, power management signaling, and training coordination. Because the sideband resides in the always-on power domain, it remains functional across mainband power states and is the sole mechanism available for link recovery when the mainband is inactive. This section describes the sideband serial packet structure and timing, the packet type taxonomy, the flow control and data integrity mechanisms, and the Performant Mode Operation feature negotiated as part of the supported optional feature set.

2.3.1 Serial Packet Structure and Timing

Physical Serial Interface. The sideband channel uses a forwarded-clock serial interface consisting of one data output pin and one clock output pin per direction per module: TXDATASB and TXCKSB on the transmit side, RXDATASB and RXCKSB on the receive side. The sideband serialization clock, forwarded to the remote die as TXCKSB, is fixed at 800 MHz regardless of the mainband data rate or link operating state; this rate applies exclusively to the serial-domain shift register that drives TXDATASB and the corresponding receive-side deserializer that samples RXDATASB. Data is

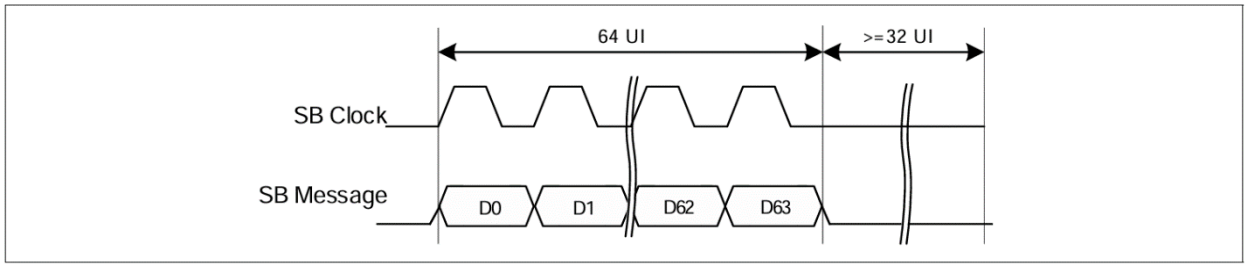


Figure 2-12: Example 64-bit Sideband Serial Packet Transfer

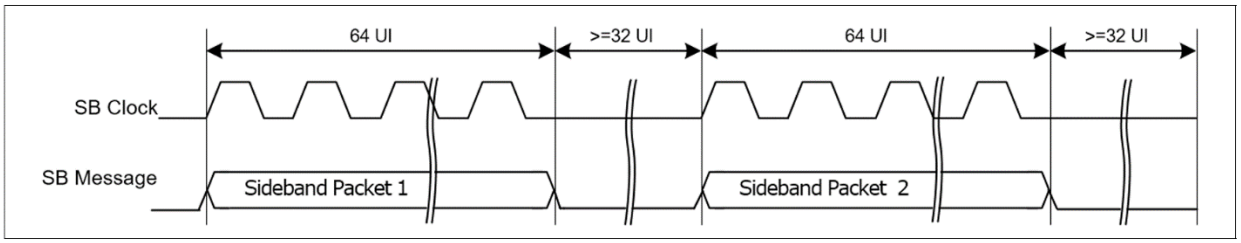


Figure 2-11: Sideband Packet Transmission: Back-to-Back

therefore transferred at a physical bit rate of 800 Mbit/s on the serial pin, such that a 64-bit packet occupies exactly 64 serial clock cycles. All digital sideband protocol logic — packet assembly and decoding, message routing, register access, and arbitration — operates at 100 MHz, derived from the 800 MHz serialization clock by a divide-by-eight stage.

Serial Packet Unit. The fundamental unit of transmission on the sideband physical interface is a 64-bit serial packet. All sideband message headers are exactly 64 bits wide and are therefore transmitted as a single serial packet. Data payloads, when present, are transmitted as a subsequent serial packet of the same 64-bit width. A 32-bit data payload is accommodated within the 64-bit data packet by padding the most significant 32 bits with zeros. This uniform 64-bit serial packet granularity means that the sideband link always operates in multiples of 64 serial bit times, irrespective of the payload width.

Bit Transmission Order. The 64-bit header is logically divided into two 32-bit phases for descriptive purposes in the specification, designated Phase 0 (bits 31:0) and Phase 1 (bits 63:32). On the serial interface, bit 0 of Phase 0 is transmitted first as the initial serial bit, followed by the remaining bits of Phase 0 in ascending order, after which bit 0 of Phase 1 is transmitted, through to bit 31 of Phase 1 as the final bit of the header serial packet. When a data payload is present, its two 32-bit phases, Phase 2 and Phase 3, follow the same ordering rule, with bit 0 of Phase 2 transmitted as the first bit of the data serial packet.

Inter-Packet Gap. Two consecutive 64-bit serial packets on the sideband interface must be separated by a minimum gap of 32 bit times during which the serial data line is held low. This mandatory gap applies between any two packets, including the gap between the header packet and the data packet of the same message. A complete sideband message without a data payload therefore occupies 64 serial bit times for the header followed by a minimum 32-bit-time low interval before the next packet may begin. A complete sideband message with a 32-bit or 64-bit data payload occupies 64 bit times for the header, 32 bit times of enforced low, 64 bit times for the data packet, and a further 32 bit times of enforced low, for a total of 192 bit times plus the post-data gap.

Sideband Performant Mode Operation. The specification defines Sideband Performant Mode Operation as a negotiable feature that modifies the inter-packet timing rule described above. When this mode is active, the mandatory 32-bit-time gap between consecutive 64-bit serial packets is eliminated, allowing packets to be transmitted back-to-back without any intervening dead time. This doubles the effective throughput of the sideband channel and is particularly relevant during the intensive sideband message traffic of the mainband training sequence. Performant Mode Operation must be supported symmetrically: a link either applies the mode in both the transmit and receive directions or does not apply it at all. Negotiation is conducted through the Sideband Feature Extensions mechanism during the mainband initialization parameter exchange. Once both dies have transmitted and received the parameter response message with the Performant Mode Operation bit set, the transmitter may begin sending packets without gaps, and the receiver must be prepared to accept them in this format. Support for this feature is included in this implementation.

Header Field Organization. The 64-bit header is organized into common fields shared across all packet types and extension fields specific to each packet category. The common fields present in every header are the five-bit opcode, which identifies the packet type and the width of any associated data payload; a three-bit source identifier and a three-bit destination identifier, which together route the packet between the Protocol Layer, D2D Adapter, Physical Layer, and Management Port Gateway; a Control Parity bit, computed as even parity across all header fields other than itself; and a Data Parity bit covering the data payload, set to zero when no payload is present.

Beyond these common fields, each packet category extends the header with type-specific content. Register access packets carry an address field, byte enables, a tag used to correlate completions with outstanding requests, a data poison indicator, and a credit return bit. Completion packets extend the header with a status field encoding the outcome of the associated request. Message packets carry a message code field and a message information field whose content is defined per message type. The detailed field layouts for each category are described in Section 2.3.2.

2.3.2 Packet Types and Message Formats

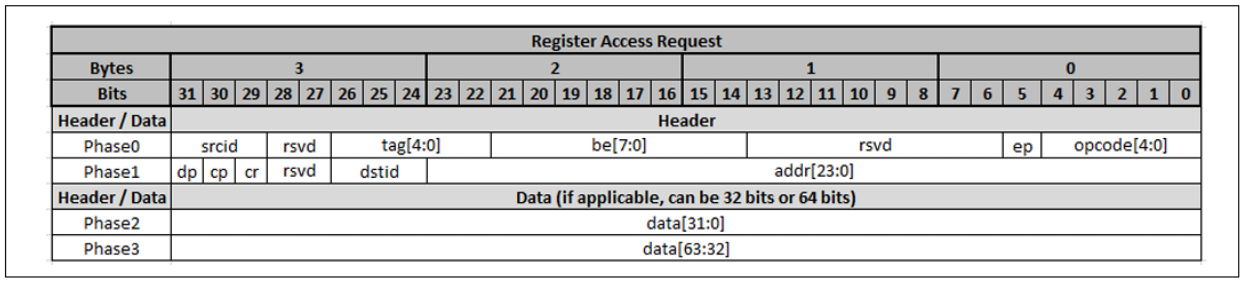
The specification defines five categories of sideband packets, distinguished by their opcode encoding, payload width, and functional purpose. A five-bit opcode field present in every packet header identifies both the packet category and the data width of any associated payload.

Table 2-3: Sideband packet opcode encodings

Opcode	Packet Type	Opcode	Packet Type
00000b	32b Memory Read	01101b	64b Configuration Write
00001b	32b Memory Write	10000b	Completion without Data
00010b	32b DMS Register Read	10001b	Completion with 32b Data
00011b	32b DMS Register Write	10010b	Message without Data
00100b	32b Configuration Read	10111b	Mgmt Port Message without Data
00101b	32b Configuration Write	11000b	Mgmt Port Message with Data
01000b	64b Memory Read	11001b	Completion with 64b Data
01001b	64b Memory Write	11011b	Message with 64b Data
01010b	64b DMS Register Read	11110b	Priority Packet
01011b	64b DMS Register Write	11111b	Priority Packet
01100b	64b Configuration Read	Others	Reserved

Register Access Packets. Register access packets initiate read or write transactions targeting configuration, memory-mapped, or DMS register spaces on the local or remote die. Six access

subtypes are defined across two data widths: 32-bit and 64-bit, each covering Memory Read, Memory Write, Configuration Read, Configuration Write, DMS Register Read, and DMS Register Write operations. Every register access transaction, whether a read or a write, produces an

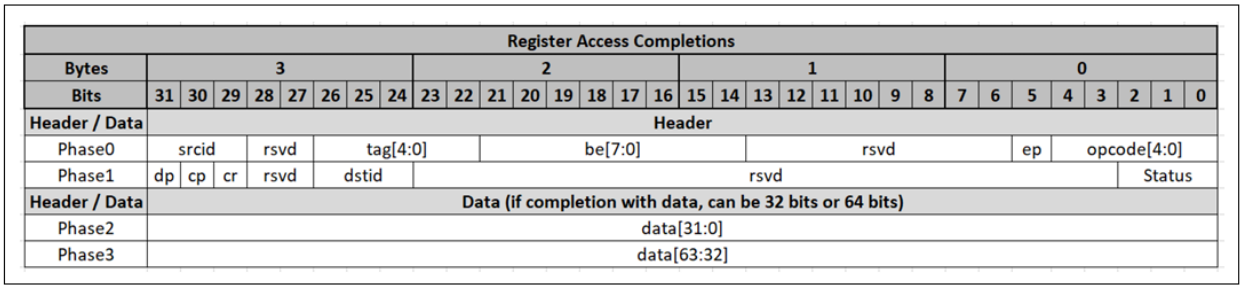


Register Access Request																															
Bytes	3								2								1								0							
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Header / Data	Header																															
Phase0	srcid			rsvd		tag[4:0]			be[7:0]								rsvd								ep	opcode[4:0]						
Phase1	dp	cp	cr	rsvd		dstid			addr[23:0]																							
Header / Data	Data (if applicable, can be 32 bits or 64 bits)																															
Phase2	data[31:0]																															
Phase3	data[63:32]																															

Figure 2-13: Format for Register Access Request

associated completion packet. The header carries the target address in a 24-bit address field, an 8-bit byte enable field, a 5-bit tag used by the requester to correlate the returning completion with the originating request, a data poison indicator, and a credit return bit. For 32-bit accesses, the two least significant address bits and the upper four byte enable bits are reserved. For 64-bit accesses, the three least significant address bits are reserved.

Completion Packets. Three completion opcodes are defined, corresponding to completions without data, with 32-bit data, and with 64-bit data. The completion header carries a 3-bit status field encoding the outcome of the associated request. A status of 000b indicates successful completion; 001b indicates an unsupported request. The tag field from the original request is echoed in the completion header, allowing the requester to match the completion to the outstanding transaction regardless of the order in which completions are returned. Control and data parity fields provide end-to-end integrity checking on the completion header and payload.



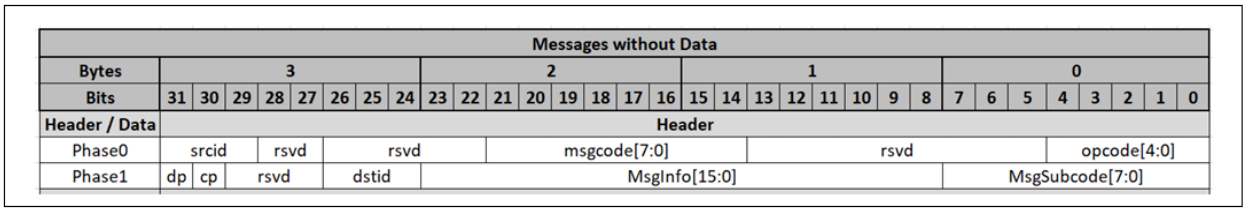
Register Access Completions																																
Bytes	3								2								1								0							
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Header / Data	Header																															
Phase0	srcid			rsvd		tag[4:0]			be[7:0]								rsvd								ep	opcode[4:0]						
Phase1	dp	cp	cr	rsvd		dstid			rsvd																						Status	
Header / Data	Data (if completion with data, can be 32 bits or 64 bits)																															
Phase2	data[31:0]																															
Phase3	data[63:32]																															

Figure 2-14: Format for Register Access Completions

Link Management Messages. The message-without-data opcode (10010b) covers two functional sub-families distinguished by their message code field. The first sub-family consists of Link Management messages, which coordinate link state transitions between the D2D Adapter and the Physical Layer on both the local and remote dies. These messages are used to negotiate entry into

and exit from the defined RDI link states, including Active, Retrain, and low-power states, and form the primary control channel through which the Adapter directs the Physical Layer during normal operation.

Link Training State Machine Messages. The second sub-family consists of Link Training State Machine messages, which coordinate the progression through the SBINIT, MBINIT, and MBTRAIN training substates. Within this family, message code B5h identifies messages originating from the local die acting as initiator, and message code BAh identifies responses from the link partner. Each

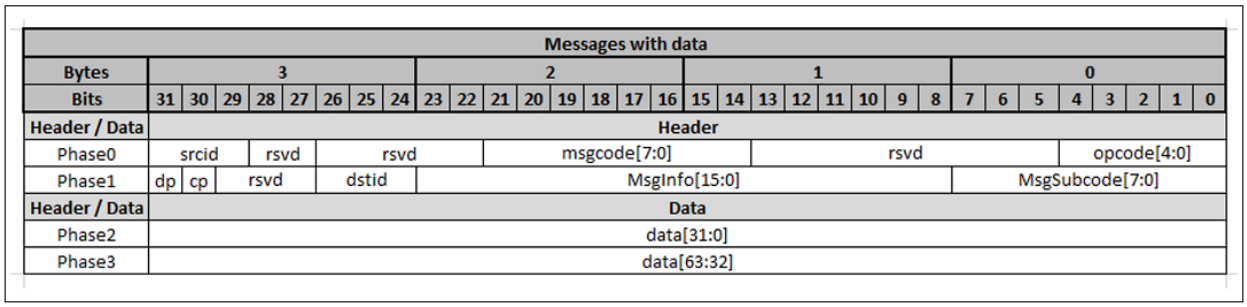


Messages without Data																													
Bytes	3								2								1								0					
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Header / Data	Header																															
Phase0	srcid				rsvd				rsvd								msgcode[7:0]								rsvd						opcode[4:0]	
Phase1	dp	cp			rsvd				dstid								MsgInfo[15:0]								MsgSubcode[7:0]							

Figure 2-15: Messages without Data

training substate defines a specific set of request and response messages with assigned subcode values. Table 7-9 of the specification enumerates the full LTSM message-without-data set, covering all handshakes from sideband initialization through the mainband repair and link speed substates.

Messages with Data. The message-with-64b-data opcode (11011b) covers parameter exchange and training-related messages that carry a 64-bit data payload in addition to the 64-bit header. These packets carry training parameters and configuration information required by selected link-management procedures. Their detailed interpretation depends on the active training substate and is described in the corresponding sections of this chapter.



Messages with data																																
Bytes	3								2								1								0							
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Header / Data	Header																															
Phase0	srcid				rsvd				rsvd								msgcode[7:0]								rsvd						opcode[4:0]	
Phase1	dp	cp			rsvd				dstid								MsgInfo[15:0]								MsgSubcode[7:0]							
Header / Data	Data																															
Phase2	data[31:0]																															
Phase3	data[63:32]																															

Figure 2-16: Format for Messages with data payloads

Management Port Messages. Management Port Messages constitute an optional packet category supporting the UCIe Management Transport protocol. Two opcode encodings are defined: Management Port Messages with data (11000b) for encapsulated transport payloads and vendor-defined gateway messages, and Management Port Messages without data (10111b) for gateway capability advertisement, credit returns, initialization completion signaling, and power management notifications. Support for this category is negotiated during sideband initialization

and is not required for compliance with the basic UCIe link operation defined by this implementation's scope.

Priority Sideband Traffic Packets. Two opcode encodings, 11110b and 11111b, are reserved for Priority Sideband Traffic Packets, which carry time-critical messages with reduced latency relative to normal sideband traffic. The priority packet header format differs in size from all other packet categories, and the specification requires that bits 5 and 6 of byte 0 be set to 11b in all priority packets to prevent single-bit error aliasing with non-priority packet types. Priority Sideband Packet Transfer is not supported by this implementation and is excluded from the design scope defined in Section 1.4.

Source and Destination Identifiers. Routing of sideband packets across the FDI, RDI, and physical sideband link is governed by the three-bit source identifier and three-bit destination identifier fields present in every non-priority packet header. The encoding of these fields differs across the three interface domains, reflecting the distinct set of agents reachable at each boundary. Table 2.x summarizes the routing scope at each interface level; the full encoding tables are reproduced from Tables 7-2 through 7-4 of the specification.

Table 2-4: Sideband routing identifier scope by interface domain

Interface	Source Identifier Represents	Destination Identifier Represents
FDI	Protocol Layer stack (Stack 0 or Stack 1)	D2D Adapter or Physical Layer
RDI	Protocol Layer, D2D Adapter, or Management Port Gateway	Local-die block or remote-die block, distinguished by the most significant destination bit
Physical Sideband Link	D2D Adapter, Physical Layer, or Management Port Gateway on the transmitting die	Corresponding functional block on the receiving die

2.3.3 Sideband Feature Extensions (SFES)

The Sideband Feature Extensions mechanism provides a structured two-stage negotiation framework within the MBINIT.PARAM training substate through which the two link partners discover and agree upon a set of optional sideband capabilities. The mechanism is designed to allow the core sideband protocol to remain interoperable between implementations with different optional feature sets, while enabling capable implementations to activate enhanced behaviors once mutual support has been confirmed.

Stage 1 — SFES Capability Advertisement. The first stage determines whether either link partner supports any sideband feature extension at all. Each die transmits the {MBINIT.PARAM configuration req} sideband message with bit 14 set to indicate SFES support, or cleared to indicate that no extensions are supported. Upon receiving the partner's configuration request, a die sets bit 14 in its own {MBINIT.PARAM configuration resp} message if and only if it both supports SFES and

received the SFES bit set in the partner's request. A die that did not set the SFES bit in its own request must not set it in its response. If both dies receive the SFES bit set in the configuration response from their respective partners across all modules, Stage 2 proceeds. If either die receives the SFES bit cleared in any module's response, the MBINIT.PARAM substate exits directly to MBINIT.CAL without a Stage 2 exchange.

Stage 2 — Individual Feature Negotiation. When Stage 1 confirms mutual SFES support, each die transmits an {MBINIT.PARAM SBFE req} message advertising individual feature capabilities and receives an {MBINIT.PARAM SBFE resp} message from the partner. This handshake occurs independently in each direction. Each feature bit in the SBFE response is set to one if and only if the corresponding bit was set in both the transmitted and the received SBFE request messages, ensuring that a feature is activated only when both dies have independently advertised support for it.

Table 2-5: SBFE feature bits in {MBINIT.PARAM SBFE req/resp}

Bit	Feature	Status in This Implementation
[0]	Management Transport Protocol (MTP)	Not supported — hardcoded 0
[1]	Sideband Performant Mode Operation (PMO)	Supported
[2]	Sideband-Only Port	Not applicable — hardcoded 0
[3]	Priority Sideband Packet Transfer (PSPT)	Not supported — hardcoded 0
[4]	L2 Sideband Power Down (L2SPD)	Not supported — hardcoded 0

Performant Mode Operation. PMO is the only SBFE feature active in this implementation. When both dies set the PMO bit in their respective SBFE request and response messages, the inter-packet gap on the sideband link is eliminated, as described in Section 2.3.1. The transmitter must adopt the gap-free transmission format after it has both sent and received an SBFE response with the PMO bit set. The receiver must be prepared to accept gap-free packets after it has transmitted its own SBFE response with the PMO bit set. From that point, PMO remains active until the LTSM enters the RESET state with the SB_MGMT_UP flag cleared. PMO can only be renegotiated on a training sequence in which the SB_MGMT_UP flag is cleared at RESET entry.

PSPT Dependency. The Priority Sideband Packet Transfer feature has a mandatory dependency on PMO: implementations advertising PSPT support must also support PMO, and both features are activated simultaneously through the same SBFE handshake. PSPT is not supported in this implementation and the PSPT bit is held at zero, consistent with the feature exclusion defined in Section 1.4.

MBINIT Stall Capability. When SFES has been successfully negotiated, the specification permits either link partner to temporarily stall progression beyond MBINIT.PARAM by transmitting a defined stall indication in the SBFE response message. During the stall interval, both link partners remain at the MBINIT.PARAM boundary until the stall condition is released. This mechanism

supports implementation-dependent activities such as firmware download through the management transport path prior to mainband training. The detailed state-machine behavior associated with this mechanism is described in Section 2.4.

Persistence Constraints. After the SB_MGMT_UP flag has been set, the SFES bit value declared in the configuration request must remain unchanged on all subsequent transitions through the MBINIT.PARAM substate until that flag is cleared. Similarly, the MTP bit value must remain stable across MBINIT.PARAM transitions once SB_MGMT_UP is set. These constraints ensure that the management transport path established during initial negotiation is not disrupted by subsequent retraining events.

2.3.4 Sideband Performant Mode Operation (PMO)

Overview and Effect. Sideband Performant Mode Operation is an optional enhancement to the baseline sideband transmission protocol that eliminates the mandatory inter-packet gap between consecutive 64-bit serial packet transfers. In the baseline sideband transmission mode described in Section 2.3.1, each 64-bit serial packet must be followed by a minimum of 32 idle bit times before the next packet may begin. When PMO is active, this dead-time requirement is removed and consecutive 64-bit serial packets may be transmitted back-to-back without any intervening idle interval. The effective sideband data throughput is thereby increased by a factor of two relative to the baseline: whereas the baseline requires a minimum of 96 UI per 64-bit packet transfer, PMO reduces the per-packet interval to exactly 64 UI. This improvement is particularly relevant during the intensive sideband message exchanges of the mainband initialization and training sequence, where multiple request-response handshakes are performed in rapid succession.

Negotiation. PMO is negotiated as part of the Sideband Feature Extensions mechanism described in Section 2.3.3. Support must be symmetric: a link either applies PMO in both the transmit and receive directions, or does not apply it at all. In a multi-module configuration, all modules must advertise the same PMO capability value. The PMO bit in the SBFE request message reflects the hardware capability and the software-enabled state of the feature; the PMO bit in the SBFE response message is set if and only if the corresponding bit was set in both the transmitted and the received SBFE request messages.

Activation Timing. The transition to gap-free transmission follows a defined sequence with distinct conditions for the transmitter and receiver. The transmitter must adopt the PMO transmission format — omitting inter-packet gaps — after it has both transmitted and received an SBFE response message with the PMO bit set to one, across all modules. The receiver must be prepared to accept gap-free packets after it has transmitted its own SBFE response with the PMO bit set to one. Because the transmitter activation condition requires receipt of the partner response while the receiver activation condition requires only the act of transmitting the local response, a timing asymmetry exists in which packets without inter-packet gaps may arrive at the receiver before the transmitter has itself confirmed PMO activation. The specification addresses this by requiring that the sideband receiver remain backward compatible and capable of accepting packets with the standard 32-UI inter-packet gap even after PMO has been negotiated. The receiver must therefore tolerate both gap-present and gap-absent packet streams during the transition interval.

Duration and Renegotiation. Once activated, PMO remains in effect across all LTSM states until the link enters the RESET state with the SB_MGMT_UP flag cleared to zero. PMO cannot be renegotiated during retraining events that leave SB_MGMT_UP set; renegotiation is only possible on a training sequence initiated from a RESET state in which SB_MGMT_UP is cleared. This persistence constraint ensures that the management transport path, if active, is not disrupted by retraining events that do not require a full link reset.

Impact on Management Port Message Transmission. PMO affects the maximum allowed transmission time for Management Port Messages with data payloads. Without PMO, a PHY transmitter must complete transmission of an MPM with Data within a maximum of 768 UI. When PMO is negotiated and active, the corresponding limit is reduced to 512 UI. The tighter bound reflects both the increased packet density enabled by gap-free transmission and the reduced arbitration latency required to allow pending link management packets to be interleaved with management payload transfers. These timing limits ensure that management traffic cannot excessively delay higher-priority link management messages during operation.

Interaction with Sideband Initialization Pattern Mode. During the sideband initialization pattern detection phase, which precedes the parameter negotiation exchange, the sideband serializer operates with fixed timing governed by the pattern generation sequence rather than by the PMO gap rule. PMO is therefore not applicable during this early initialization interval and the gap-free format must not be used before the SFES negotiation has concluded with the PMO bit confirmed in both directions.

Implementation Support. PMO is included within the scope of this work and is fully supported by the proposed digital PHY architecture. The architectural mechanisms used to negotiate and apply PMO are presented in Chapter 3.

2.4 Link Training State Machine (LTSM) Specification

The Link Training State Machine governs the complete lifecycle of a UCIe physical link from initial power-on through active data transfer, low-power operation, retraining, and error recovery. The LTSM defines a set of named states, each with specific entry conditions, physical layer behaviors, and exit criteria, together with a set of transition rules that connect the states into a coherent link management sequence. This section describes the state definitions, the primary and exception transition paths, the error handling mechanism, and the watchdog timer that bounds the duration of any individual state or substate.

2.4.1 State Definitions and Transition Rules

State Definitions. The table summarizes the nine states of the LTSM as defined in Table 4-6 of the specification.

Table 2-6: LTSM state definitions

State	Description
RESET	Entry state following primary reset or exit from TRAINERROR. PLLs stabilize and initialization preconditions are established.
SBINIT	Sideband initialization. The sideband channel is activated, clock detection and pattern validation are performed, and the out-of-reset message is exchanged.
MBINIT	Mainband initialization at the lowest supported data rate. On-die calibration is performed, followed by interconnect repair where applicable and parameter negotiation.
MBTRAIN	Mainband training. The link is brought to the highest negotiated data rate and die-to-die training is performed to align the forwarded clock with respect to data across all active lanes.
LINKINIT	Protocol-level link initialization. The D2D Adapter and link management messages required to establish the RDI Active state are exchanged.
ACTIVE	Operational state in which mission-mode Flit data transfer is performed.
L1 / L2	Power management states providing link power savings beyond dynamic clock gating. L1 is a shallow power state from which the link exits through MBTRAIN. L2 is a deep power state from which the link exits through RESET.
PHYRETRAIN	Runtime retrain state. Entered from ACTIVE when a retrain condition is detected, enabling renegotiation of link parameters without a full link reset.
TRAINERROR	Error state entered upon detection of a fatal or non-fatal fault during any phase of training or operation. The link exits TRAINERROR to RESET after completing the required error handshake.

Primary Forward Transition Sequence. Under normal initialization, the LTSM advances sequentially through RESET, SBINIT, MBINIT, MBTRAIN, LINKINIT, and ACTIVE. Entry into each state requires the preceding state to signal completion through its defined exit handshake or done condition. The link does not skip any state in this forward sequence during initial bring-up. Figure 2.x reproduces the LTSM diagram from Figure 4-33 of the specification.

RESET State Requirements. The physical layer must remain in RESET for a minimum of four milliseconds upon every entry, to allow PLL stabilization and initialization preconditions to be met. The minimum conditions for RESET exit require that power supplies are stable and the sideband clock is running at 800 MHz. In addition, either the mainband clock must be set to the slowest

supported I/O data rate and the local firmware must not be holding the PHY in RESET and a link training trigger must have occurred, or the Management Transport conditions for sideband-only bring-up must be satisfied. While in RESET, all Data, Valid, Clock, and Track transmitters are tri-stated, their receivers may be disabled, and the sideband transmitter is held low unless the SB_MGMT_UP flag is set.

L1 and L2 Distinctions. L1 is a shallow power management state that preserves the negotiated link parameters and allows rapid resumption of data transfer. Upon L1 exit, the LTSM transitions directly to MBTRAIN and re-enters the training sequence at the SPEEDIDLE substate, bypassing the voltage reference calibration substates that were already performed during initial bring-up. L2 is a deeper power state after which the full link initialization sequence must be repeated; the L2 exit path leads to RESET rather than to MBTRAIN.

PHYRETRAIN. PHYRETRAIN is entered from ACTIVE when a retrain condition is detected — either by the local Adapter, by detection of a PHY framing error, at the request of the remote die, or as a result of link quality assessment during the MBTRAIN.LINKSPEED substate. During PHYRETRAIN, the two dies exchange a retrain handshake that negotiates the re-entry point for the subsequent MBTRAIN visit. Three re-entry points are defined: MBTRAIN.TXSELFCAL for cases where no lane repair is required, MBTRAIN.SPEEDIDLE for cases where a link speed change is needed, and MBTRAIN.REPAIR for cases where width degradation is required. Upon completion of the PHYRETRAIN handshake, the LTSM transitions to MBTRAIN at the resolved re-entry point.

TRAINERROR Entry and Handshake. TRAINERROR may be entered from any state except RESET. The mechanism for entering TRAINERROR differs by origin state. From SBINIT and PHYRETRAIN, the transition to TRAINERROR is direct upon detection of the error condition or expiry of the watchdog timer. From MBINIT, MBTRAIN, and LINKINIT, the entry to TRAINERROR requires completion of a two-way sideband handshake defined in Section 4.5.3.8 of the specification, through which both dies acknowledge the error condition before the state is changed. Entry into TRAINERROR depends on the state from which the error originates. Some states transition directly upon timeout or fault detection, whereas others require a coordinated sideband handshake between the two dies before the state change is committed. Upon completion of the required error handling procedure, the LTSM always returns to RESET.

Watchdog Timer. A per-state eight-millisecond watchdog timer is active during SBINIT, MBINIT, MBTRAIN, LINKINIT, and PHYRETRAIN. The timer is restarted with a fresh budget on every substate transition, so each individual substate within MBINIT and MBTRAIN is subject to its own eight-millisecond limit rather than sharing a single budget across the entire parent state. A timeout in SBINIT or following a second timeout in any active training state causes a direct transition to TRAINERROR. A timeout in MBINIT, MBTRAIN, or LINKINIT triggers the TRAINERROR entry handshake before the state transition completes.

Substate Structure. MBINIT and MBTRAIN each comprise a sequence of defined substates that are executed in order under the control of the parent state. MBINIT substates cover sideband feature negotiation, on-die calibration, clock and valid lane repair validation, lane reversal detection, and mainband lane repair. MBTRAIN substates cover voltage reference calibration, link speed

negotiation, transmitter self-calibration, receiver clock calibration, phase interpolator centering, voltage reference training for valid and data lanes, receiver deskew, and link speed stability verification. The substate sequences, their entry conditions, and their individual exit criteria are described in Sections 2.4.2 and 2.4.3 respectively.

2.4.2 MBINIT Sub-states

MBINIT initializes the mainband interface at the lowest supported data rate of 4 GT/s. In the Standard Package, no physical lane repair is possible; the six substates within MBINIT are therefore used to negotiate link parameters, perform on-die calibration, verify the integrity of the clock and valid lanes, detect and correct lane reversal, and establish the functional lane map through width degradation if required. The substates execute in the fixed sequence shown in Figure 4-34 of the specification: PARAM → CAL → REPAIRCLK → REPAIRVAL → REVERSALMB → REPAIRMB. Each substate is bounded by the shared eight-millisecond watchdog defined in Section 2.4.1.

MBINIT.PARAM. The parameter exchange substate negotiates the operating configuration that will govern the remainder of link bring-up. While mainband transmitters remain tri-stated and receivers may be disabled, the two dies exchange {MBINIT.PARAM configuration req} and {MBINIT.PARAM configuration resp} sideband messages, each carrying a structured data payload. The parameters communicated in the request include the transmitter voltage swing, the maximum supported data rate, the clock mode preference for operation at or below 32 GT/s, the Standard Package Module Width capability bit, and the SFES capability bit. The responding die sets the maximum common data rate as the minimum of the locally supported and remotely advertised values, and reflects back only the capabilities it shares. Additional parameters governing the Parity Feature and TARR capability follow the same agreement rule. If SFES capability is confirmed in the configuration exchange, a second pair of messages — the {MBINIT.PARAM SBFE req} and {MBINIT.PARAM SBFE resp} — immediately follows to negotiate individual sideband feature extension bits, including PMO, as described in Section 2.3.3. If an MBINIT stall condition is signalled in the SBFE response, both dies hold at the PARAM boundary until the stall is released. Upon completion of the parameter exchange, the negotiated values are stored and used throughout the remainder of link initialization and subsequent link operation unless renegotiated during retraining.

MBINIT.CAL. The calibration substate provides a time window in which each die performs independent on-die initialization activities whose completion must be coordinated with the link partner before training can proceed. The activities performed during CAL are implementation-defined and are not mandated by the specification beyond their existence. Typical activities include analog bias stabilization and other implementation-specific calibration procedures. The substate uses a three-way handshake: an initialization request and response confirm that both dies are ready to begin, calibration proceeds locally, and a completion request and response confirm that both dies have finished. Mainband transmitters remain tri-stated and receivers may remain disabled throughout CAL.

MBINIT.REPAIRCLK. The clock repair validation substate verifies the integrity of the forwarded clock lanes at the 4 GT/s operating rate. The transmitter drives a clock pattern on the Phase-1 and Phase-2 clock lanes while holding data and valid lanes inactive, and the receiver evaluates the received

clock pattern against the expected waveform. The substate uses an initialization handshake, followed by a pattern transmission and comparison interval, followed by a result exchange in which each die reports its local clock lane detection outcome to the partner, and concludes with a completion handshake. For the Standard Package, a detection failure on either clock lane is a non-recoverable condition that causes the substate to exit to TRAINERROR, because no redundant clock lanes exist in the Standard Package.

MBINIT.REPAIRVAL. The valid repair validation substate verifies the integrity of the Valid lane using the same handshake structure as REPAIRCLK. The transmitter drives the Valid training pattern defined in Section 2.2.5 on the Valid lane while holding the clock in the static differential-low condition. The receiver evaluates the received Valid pattern using the valid-lane comparison mode. The local detection outcome is exchanged with the partner, and a completion handshake closes the substate. A detection failure causes the substate to exit to TRAINERROR, because no Valid lane redundancy exists in the Standard Package.

MBINIT.REVERSALMB. The lane reversal detection substate determines whether the physical data lane connections between the two dies are ordered in the forward or reversed orientation, using the mechanism described in Section 2.2.3. The substate transmits the Per-Lane ID pattern across all active data lanes, exchanges per-lane detection results through sideband messages, and applies lane reversal on the transmitter if the majority criterion indicates a reversed connection. If the first orientation produces a majority of successful detections, no reversal is applied. If it does not, lane reversal is applied and the pattern sequence is repeated once. If neither attempt yields a majority of successful lane detections, the substate exits to TRAINERROR. A successful outcome in either direction is followed by a completion handshake before REVERSALMB exits to REPAIRMB.

MBINIT.REPAIRMB. The mainband repair substate performs the initial width degradation assessment using a data-to-clock point test at 4 GT/s. Each die transmits the PRBS pattern on all active data lanes with correct Valid framing and the forwarded clock, and the receiver evaluates the per-lane comparison results. The substate begins with a readiness handshake, runs the point test, and then each die communicates its locally determined functional lane map encoding — using the three-bit encodings defined in Section 2.2.4 — to its partner through the apply-degrade sideband message exchange. The receiver of each die is then configured to match the lane map reported by the remote partner. If the initial test resolves to a valid lane map on both sides, a completion handshake closes the substate and the LTSM transitions to MBTRAIN with the agreed lane configuration. If no valid lane map can be found within the retry procedure defined by the specification, the substate exits to TRAINERROR.

2.4.3 MBTRAIN Sub-states

MBTRAIN elevates the mainband link from the 4 GT/s initialization rate used during MBINIT to the highest data rate negotiated during MBINIT.PARAM and performs the die-to-die calibration required to establish reliable timing and voltage alignment between the forwarded clock and the received data eye across all active lanes at the target operating speed. The state comprises thirteen sequential substates whose combined objective is to bring the link to a stable, calibrated operating condition. The normal entry path from MBINIT begins at VALVREF and advances linearly to

LINKINIT; alternative entry points are defined for L1 power management exit, PHYRETRAIN re-entry, and post-training width repair.

Table 2-7: MBTRAIN substate sequence and objectives.

Substate	Primary Objective
VALVREF	Coarse receiver Vref calibration for the Valid lane
DATAVREF	Coarse receiver Vref calibration for all active data lanes
SPEEDIDLE	Link speed negotiation and PLL clock rate transition
TXSELFCAL	Transmitter self-calibration
RXCLKCAL	Receiver clock phase alignment and I/Q correction
VALTRAINCENTER	Transmitter phase interpolator centering for the Valid lane
VALTRAINVREF	Fine receiver Vref calibration for the Valid lane
DATATRAINCENTER1	First-pass transmitter PI centering for all active data lanes
DATATRAINVREF	Fine receiver Vref calibration for all active data lanes
RXDESKEW	Per-lane receiver deskew and equalization preset tuning
DATATRAINCENTER2	Second-pass transmitter PI centering for all active data lanes
LINKSPEED	Link stability verification at the final operating rate
REPAIR	Post-training width degradation and lane remapping

MBTRAIN.VALVREF. The Valid lane voltage reference calibration substate establishes the initial receiver decision threshold for the Valid lane before any high-speed signaling begins. This coarse calibration is performed at the 4 GT/s rate carried forward from MBINIT and provides a starting point for the fine calibration performed later in VALTRAINVREF. The substate completes through a coordinated local-and-partner handshake and exits to DATAVREF.

MBTRAIN.DATAVREF. The data lane voltage reference calibration substate performs the equivalent coarse Vref initialization for all active data lanes. As with VALVREF, this calibration establishes an initial decision threshold that is subsequently refined after clock-to-data alignment has been completed. The substate exits to SPEEDIDLE upon coordinated completion.

MBTRAIN.SPEEDIDLE. The speed negotiation substate coordinates the transition of both dies from the 4 GT/s initialization rate to the maximum mutually supported data rate agreed upon during MBINIT.PARAM. The two dies exchange speed transition messages through the sideband, each confirming readiness before the PHY transitions to the negotiated operating clock rate. The new rate must be one of the data rates supported by both dies as advertised during MBINIT.PARAM. If the target rate cannot be achieved and no lower valid rate remains available, the substate exits to TRAINERROR. SPEEDIDLE is also the re-entry point when LINKSPEED requests a speed downgrade

due to link instability at the current rate, and is the entry point when MBTRAIN is entered following an L1 power management exit.

MBTRAIN.TXSELFCAL. The transmitter self-calibration substate performs implementation-specific calibration of the transmit drivers at the new operating data rate, although the exact calibration procedure is implementation-dependent. Typical objectives include impedance matching and slew rate adjustment. The substate is coordinated through a local-and-partner handshake. In addition to being entered sequentially from SPEEDIDLE, TXSELFCAL is a defined PHYRETRAIN re-entry point and follows successful completion of the post-training REPAIR substate, ensuring transmitter calibration is refreshed whenever the active lane configuration changes.

MBTRAIN.RXCLKCAL. The receiver clock calibration substate locks the receiver sampling clock to the forwarded clock from the transmitter and corrects any in-phase/quadrature phase offset. The transmitter drives a clock pattern on the Phase-1 and Phase-2 clock lanes and on the Track lane while the receiver measures the phase relationship and adjusts its sampling alignment accordingly. If the required phase correction exceeds the physical adjustment range of the transmit clock driver, the substate exits to TRAINERROR.

MBTRAIN.VALTRAINCENTER. The Valid lane phase interpolator centering substate adjusts the transmitter PI phase on the Valid lane to horizontally center the transmitted Valid signal within the receiver's sampling window. The transmitter drives the Valid training pattern while the receiver's valid lane comparator evaluates detection quality across a range of phase codes. The substate uses a coordinated test-and-report sequence communicated through the sideband to converge on the optimal PI code for the Valid lane.

MBTRAIN.VALTRAINVREF. Following PI centering, the Valid lane voltage reference training substate fine-tunes the receiver Vref for the Valid lane using the optimized transmitter clock phase established in VALTRAINCENTER. The refined Vref improves the noise margin of the Valid framing mechanism and provides a more accurate decision threshold for the remainder of the training sequence and for mission-mode operation.

MBTRAIN.DATATRAINCENTER1. The first-pass data lane PI centering substate performs initial clock-to-data alignment for all active data lanes simultaneously, using the PRBS training pattern defined in Section 2.2.5. The transmitter sweeps PI phase codes per lane while the receiver per-lane comparator evaluates the sampled pattern quality at each point. The resulting per-lane PI codes provide a coarse centering of the data eye relative to the forwarded clock. DATATRAINCENTER1 is also the re-entry point of the deskew loopback arc described under RXDESKEW.

MBTRAIN.DATATRAINVREF. After the first-pass PI centering, the data lane voltage reference training substate re-calibrates the receiver Vref for each active data lane using the improved transmitter clock alignment. Fine Vref calibration at this stage provides better noise margin estimates for subsequent deskew and the second-pass PI centering.

MBTRAIN.RXDESKEW. The receiver deskew substate corrects the lane-to-lane skew arising from physical interconnect length differences and optimizes transmitter equalization presets for each active lane. Per-lane deskew codes are swept and evaluated through a pattern-based

measurement sequence. If the optimization reveals that the lane-to-lane timing relationships have changed significantly relative to the assumptions made during DATATRAINCENTER1, the substate may request a loopback to DATATRAINCENTER1 to repeat the PI centering with the corrected deskew applied. This loopback arc may execute up to four times; if the partner requests a fifth loopback after the limit has been reached, the substate exits to TRAINERROR.

MBTRAIN.DATATRAINCENTER2. The second-pass data lane PI centering substate fine-tunes the transmitter clock alignment for each active data lane after deskew has been applied. With the per-lane timing corrections from RXDESKEW in place, the second centering pass achieves a more accurate alignment than the first pass and establishes the final PI code set used during mission-mode operation.

MBTRAIN.LINKSPEED. The link speed verification substate confirms that the link operates stably at the final negotiated data rate under the final operating configuration. A PRBS pattern is transmitted on all active lanes for a defined measurement interval and the per-lane comparison results are evaluated. Four exit paths are defined. If all active lanes pass the stability test, the substate signals completion and the LTSM transitions to LINKINIT. If lane instability is detected but the number of failing lanes is compatible with a width-reduced configuration, the substate requests transition to REPAIR. If the link is unstable at the current data rate but a lower rate is available, the substate requests transition back to SPEEDIDLE for a speed downgrade, after which the full calibration sequence from TXSELFCAL is repeated. If a PHYRETRAIN condition is detected during LINKSPEED, the substate initiates the PHYRETRAIN handshake before exiting MBTRAIN.

MBTRAIN.REPAIR. The post-training width degradation substate applies the width reduction mechanism described in Section 2.2.4 when LINKSPEED has determined that one or more lanes are not viable at the operating data rate. The per-lane results from LINKSPEED determine the local lane map encoding, which is communicated to the partner die through the sideband apply-degrade message exchange. Each die configures its receiver to match the partner's reported lane map. If a valid degraded configuration can be agreed upon, the substate completes successfully and the LTSM returns to TXSELFCAL, ensuring that transmitter self-calibration is refreshed at the reduced lane configuration before the training sequence resumes. If no feasible lane map exists, the substate exits to TRAINERROR.

2.4.4 LINKINIT, ACTIVE, PHYRETRAIN, and TRAINERROR

LINKINIT. Following the completion of MBTRAIN, the LTSM enters LINKINIT, in which the D2D Adapter and Protocol Layer bring-up exchanges required to establish the RDI Active state are performed over the sideband channel. The Physical Layer's role in this state is to transport the sideband messages exchanged between the local Adapter and the remote link partner without interpreting their protocol-layer content. The exchanges conducted during LINKINIT constitute Stage 3 of the link initialization sequence defined by the specification. In Part 1 of Stage 3, the Adapter advertises protocol capability information — including supported Flit formats and multi-protocol capability — through pairs of capability advertisement and finalization messages. In Part 2, if multi-protocol capability has been negotiated, additional protocol-specific capability exchanges are performed. Both parts are subject to a collective eight-millisecond timeout

measured from the point at which the RDI transitions to Active state; a stall message received from the remote partner resets this timer. Upon completion of the capability exchanges, the Active entry handshake is performed: each die sends a {LinkMgmt.Adapter.Req.Active} sideband message and responds to the partner's equivalent with a {LinkMgmt.Adapter.Rsp.Active} message, confirming that the local receiver is ready to accept Flits. Once both dies have sent and received the Active response, the Physical Layer transitions the RDI to Active state and the LTSM advances to ACTIVE. Upon entering ACTIVE, the specification initializes the sideband register-access credit mechanism to its defined starting state, establishing the initial flow control condition for register access transactions over the sideband.

ACTIVE. ACTIVE is the mission-mode state of the LTSM in which mainband Flit data transfer proceeds continuously at the negotiated data rate and lane configuration. All calibration parameters established during MBTRAIN remain active throughout mission-mode operation, including the transmitter and receiver timing and voltage settings and the negotiated active lane configuration. The sideband channel remains available for register access transactions, link management messages, and power management coordination throughout the duration of the Active state. Mainband clock gating and idle insertion follow the Valid framing rules described in Section 2.2.1. Four primary exit paths are defined from ACTIVE. First, if the D2D Adapter requests a retrain through the RDI, the LTSM exits to PHYRETRAIN after completing the required RDI stall handshake. Second, if the Adapter requests a power management transition, the LTSM exits to L1 or L2 depending on the requested depth. Third, if a Retrain request is received from the remote link partner via the sideband, the LTSM enters PHYRETRAIN. Fourth, a fatal link condition signalled through the RDI — including LinkError, LinkReset, or Disabled — causes the LTSM to exit to TRAINERROR. For all graceful exits, the specification requires that both the physical layer receive-active request and the Adapter receive-active status signals de-assert before the state transition is committed, ensuring that no Flits are in flight when the link exits the Active state.

PHYRETRAIN. PHYRETRAIN is a runtime state entered from ACTIVE when retraining the mainband link is required without performing a full link reset. Trigger sources include an explicit Retrain request from the local Adapter, detection of a PHY-level framing error, receipt of a retrain request from the remote die over the sideband, or a determination made by MBTRAIN.LINKSPEED that link parameters have changed since the previous training session. Upon entry, PHYRETRAIN performs a four-step sideband handshake in which each die advertises a three-bit retrain encoding representing its assessment of the minimum re-entry point required for the upcoming MBTRAIN session. Three re-entry encodings are defined: TXSELFCAL for cases requiring only transmitter recalibration, SPEEDIDLE for cases requiring a speed renegotiation, and REPAIR for cases where width degradation must be applied. The resolved re-entry encoding is determined as the higher-priority of the two advertised values, where REPAIR takes priority over SPEEDIDLE, which takes priority over TXSELFCAL. Each die's local encoding is based on the locally detected link status and the feasibility of further width degradation given the current lane map. Once both dies have exchanged their encodings and a resolution has been agreed upon, PHYRETRAIN exits to MBTRAIN at the resolved re-entry substate. If MBTRAIN.LINKSPEED subsequently determines that all lanes pass the stability test but detects that link parameters have changed since the PHYRETRAIN was

initiated, LINKSPEED requests a return to PHYRETRAIN for a further round of retrain negotiation rather than advancing to LINKINIT.

TRAINERROR. TRAINERROR is the error recovery state entered whenever a fatal or non-fatal fault is detected during any phase of link training or operation. As described in Section 2.4.1, the entry mechanism varies by origin state: direct transition for faults in SBINIT and PHYRETRAIN, and a coordinated two-way sideband handshake for faults in MBINIT, MBTRAIN, and LINKINIT. While in TRAINERROR, mainband transmitters are placed in an electrically idle condition and the sideband remains active to permit the error handshake to complete. Implementations typically record diagnostic information associated with the detected fault, including the substate in which the fault occurred, for subsequent software analysis. TRAINERROR is the only state from which the LTSM does not exit to another training state; its sole exit path leads unconditionally to RESET, after which the full link initialization sequence begins again from SBINIT.

2.5 Raw Die-to-Die Interface (RDI) Specification

The Raw Die-to-Die Interface defines the boundary between the Physical Layer and the D2D Adapter within a single die. Its primary function is to abstract the internal state of the Physical Layer — including link training progress, power management conditions, and error events — into a set of well-defined signals and state encodings that the Adapter can act upon without knowledge of the underlying training procedure. The RDI carries three categories of information: link state control and status, clock and wake management, and framing-level error indications. The data path between the Physical Layer and the Adapter is also exposed through the RDI, at a width of at least one byte per physical lane per module. The signal naming convention follows the direction of drive: signals prefixed with pl_ are driven by the Physical Layer toward the Adapter, and signals prefixed with lp_ are driven by the Adapter toward the Physical Layer.

2.5.1 RDI Signal List and State Machine

Signal List. The next table lists the RDI control and status signals relevant to the digital Physical Layer implementation. Data path signals are excluded from this table as their width is a function of the negotiated lane configuration and are described in the context of the mainband data path.

Table 2-8: RDI control and status signal list

Signal	Direction	Width	Description
lp_state_req	Adapter → PHY	4	Adapter request for link state transition. Encodings: NOP (0000b), Active (0001b), L1 (0100b), L2 (1000b), LinkReset (1001b), Retrain (1011b), Disabled (1100b).
pl_state_sts	PHY → Adapter	4	Physical Layer current state status. Encodings: Reset (0000b), Active (0001b), Active.PMNAK (0011b), L1 (0100b), L2 (1000b), LinkReset (1001b), LinkError (1010b), Retrain (1011b), Disabled (1100b).
pl_inband_pres	PHY → Adapter	1	Asserted when the Physical Layer has completed parameter negotiation with the remote partner and is

			ready for the Adapter to request the Active state. Remains asserted through Active, Retrain, L1, and L2; de-asserts during Reset, LinkReset, Disabled, and LinkError.
pl_clk_req	PHY → Adapter	1	Physical Layer request to the Adapter to remove clock gating. Must be acknowledged by lp_clk_ack before the Adapter may clock-gate its logic.
lp_clk_ack	Adapter → PHY	1	Adapter acknowledgment that clock gating has been removed. At least one clock cycle must separate pl_clk_req assertion from lp_clk_ack assertion.
lp_wake_req	Adapter → PHY	1	Adapter wake-up request, used before state transitions to ensure the Physical Layer clocks are active before lp_state_req changes.
pl_wake_ack	PHY → Adapter	1	Physical Layer acknowledgment of the Adapter wake-up request.
pl_stallreq	PHY → Adapter	1	Physical Layer request to the Adapter to flush all in-flight Flits and inhibit new ones, required before graceful state transitions out of Active.
lp_stallack	Adapter → PHY	1	Adapter acknowledgment that Flits are aligned and stalled.
pl_error	PHY → Adapter	1	Physical Layer indication of a receive-path framing error. Pipeline-matched with the receive data path. Triggers Retrain when detected during Active state.
pl_trainerror	PHY → Adapter	1	Physical Layer indication that a training error has been detected.
lp_linkerror	Adapter → PHY	1	Adapter-initiated link error indication. The Physical Layer must propagate this to the RDI state machine and transition to LinkError.
pl_speedmode	PHY → Adapter	3	Current operating data rate of the link.
pl_max_speedmode	PHY → Adapter	4	Maximum negotiated data rate as established during MBINIT.PARAM.
pl_lnk_cfg	PHY → Adapter	2	Current link width configuration, reflecting the active lane map following any width degradation.
pl_phyinrecenter	PHY → Adapter	1	Asserted when the Physical Layer is performing recentering; the Adapter must not initiate power management while this signal is asserted.

RDI State Machine. The RDI state machine defines the set of states and transitions that the Physical Layer exposes to the Adapter over the pl_state_sts signal. The state machine has nine defined states: Reset, Active, Active.PMNAK, Retrain, L1, L2, LinkReset, LinkError, and Disabled. Figure 2.x reproduces the RDI state diagram from Figure 10-5 of the specification.

The Physical Layer enters Reset following a domain reset or upon exit from any TRAINERROR condition. The Reset state is held until the Physical Layer has completed the link training sequence through LINKINIT. The transition from Reset to Active requires the Active Entry Handshake described below.

Active is the operational state in which Flit transfer proceeds on the mainband. Transitions out of Active to Retrain, L1, L2, LinkReset, and Disabled all require the stall handshake to complete — pl_stallreq asserted and lp_stallack received — before the state transition is committed, ensuring that no Flits are left partially transmitted. The transition to LinkError may occur immediately upon detection of a fatal condition, without waiting for the stall handshake, though both pl_rx_active_req and lp_rx_active_sts must still de-assert before the state transitions away from LinkError.

Active.PMNAK is a transient sub-state entered when a power management entry request is rejected by the remote link partner. The Adapter must return to requesting Active after observing Active.PMNAK before retrying the power management request.

Retrain is entered when either the local Adapter or the remote link partner requests retraining. During Retrain, the mainband data path is suspended and the LTSM re-enters PHYRETRAIN, as described in Section 2.4.4. Upon successful completion of the subsequent MBTRAIN session, the RDI returns to Active through a new Active Entry Handshake.

L1 and L2 are the power management states described in Section 2.4.1. L1 exit returns the RDI to Active through a retraining sequence entering MBTRAIN at SPEEDIDLE. L2 exit causes a full RESET-to-Active sequence.

LinkReset is entered when the Adapter requests a link reset through lp_state_req or when a {LinkMgmt.LinkReset.Req} sideband message is received from the remote partner. Following the completion of the LinkReset sideband handshake, the RDI returns to Reset. LinkError is entered on fatal fault conditions from any RDI state, including both lp_linkerror assertion and internally detected physical layer faults. Disabled is entered when the Adapter explicitly requests link shutdown; it is a terminal state from which only a power cycle or domain reset provides recovery.

Active Entry Handshake. Every transition from Reset to Active requires a coordinated six-step handshake. Following completion of the LTSM training sequence, the Physical Layer asserts pl_inband_pres, which informs the Adapter that parameter negotiation with the remote partner is complete. The Adapter performs the lp_wake_req handshake to activate its clocks and then sets lp_state_req to Active. Upon sampling lp_state_req equal to Active, the Physical Layer transmits the {LinkMgmt.RDI.Req.Active} sideband message to the remote Physical Layer and waits for the corresponding response. The {LinkMgmt.RDI.Rsp.Active} response may only be sent after the responding Physical Layer has itself sampled lp_state_req equal to Active from its own local Adapter. Once both Physical Layers have sent and received the Active response message, pl_state_sts is set to Active on both sides simultaneously, completing the handshake. These steps constitute the Active Entry Handshake and must be executed on every entry to the Active state, including entries following Retrain, L1, and L2 exits.

2.5.2 RDI State Transitions and PHY Control Functions

Transition Triggering Rules. Each RDI state transition is conditional on both the current state and the nature of the triggering event. The specification defines in Table 10-4 which requests the Physical Layer is required to act upon in each state, and which it must silently ignore. Table 2.x summarizes the subset of transitions relevant to this implementation.

Table 2-9: Requests Considered in Each State by Lower Layer

Request (Row) Versus Status (Column)	Reset	Active	L1	LinkReset	Retrain	Disable	L2	LinkError
NOP	Yes	Ignore	Ignore	Ignore	Ignore	Ignore	Ignore	Ignore
Active	Yes[a]	Ignore[b]	Yes	Yes	Yes	Yes	Yes	Yes
L1	Ignore	Yes	Ignore	Ignore	Ignore	Ignore	Ignore	Ignore
LinkReset	Yes[a]	Yes	Yes	Ignore	Yes	Ignore	Yes	Ignore
Retrain	Ignore	Yes	Ignore	Ignore	Ignore	Ignore	Ignore	Ignore
Disable	Yes[a]	Yes	Yes	Yes	Yes	Ignore	Yes	Ignore
L2	Ignore	Yes	Ignore	Ignore	Ignore	Ignore	Ignore	Ignore
LinkError (sideband wire)	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

a. Requires request transition from NOP
b. If the Status is Active.PMNAK, then the Lower Layer transitions to Active upon sampling the Active Request.

Active → Retrain. The Physical Layer transitions pl_state_sts to Retrain when any of the following conditions is detected while in the Active state: the Adapter asserts lp_state_req equal to Retrain; a receive-path framing error is detected and signalled on pl_error; the LTSM internally determines that retraining is required; or a {LinkMgmt.RDI.Req.Retrain} sideband message is received from the remote link partner. All Retrain entry paths require the stall handshake to be completed first: the Physical Layer asserts pl_stallreq and waits for lp_stallack before initiating the sideband retrain exchange and directing the LTSM to enter PHYRETRAIN. Upon completion of the subsequent MBTRAIN session and a new Active Entry Handshake, pl_state_sts returns to Active.

Active → L1 and Active → L2. Power management state entry follows a symmetric two-way handshake between the two Physical Layers over the sideband. The Adapter must ensure that all its internal link state machines have reached the corresponding power management state, with no Flits or acknowledgments pending, before asserting lp_state_req equal to L1 or L2. The Physical Layer initiates the handshake by transmitting a {LinkMgmt.RDI.Req.L1} or {LinkMgmt.RDI.Req.L2} sideband message to the remote Physical Layer and awaiting the corresponding response. Upon receiving a PM Request sideband message from the remote partner, the Physical Layer must respond within two microseconds. If the local Adapter has also requested the same power management state, the Physical Layer responds with the PM Response message and transitions pl_state_sts to the power management state. If the local Adapter has not yet requested power management and one microsecond elapses after receiving the PM Request, the Physical Layer responds with a {LinkMgmt.RDI.Rsp.PMNAK} message. Power management entry is considered complete once both Physical Layers have received a valid PM Response. Upon entry, Data, Valid,

Clock, and Track transmitters are tri-stated and receivers may be disabled; the sideband logic and its associated clock generation must remain active throughout L1.

Active.PMNAK. When a Physical Layer receives a PMNAK response to its own PM Request, it transitions pl_state_sts to Active.PMNAK. This sub-state signals to the Adapter that power management entry was rejected by the remote partner. The Physical Layer is permitted to retry power management entry at least two microseconds after receiving the PMNAK, provided all PM entry conditions remain satisfied. There is no transition from Active.PMNAK to L1 or L2 directly; the Adapter must return to requesting Active before a subsequent power management attempt may proceed.

L1 Exit. PM exit from L1 is initiated either by the local Adapter asserting lp_state_req equal to Active, or by receipt of a {LinkMgmt.RDI.Req.Active} sideband message from the remote Physical Layer. In either case, the Physical Layer transitions pl_state_sts to Retrain, performs the pl_clk_req handshake with the Adapter to restore clock activity, and directs the LTSM to enter MBTRAIN at the SPEEDIDLE substate. Following the completion of MBTRAIN and LINKINIT, the Active Entry Handshake re-establishes the Active state on both sides. The timeout counter for the Active Entry sideband message handshake begins only after the LTSM has reached LINKINIT.

L2 Exit. PM exit from L2 follows the same structure as L1 exit in terms of triggering, but the re-entry path leads through the Reset state rather than Retrain. A full link initialization sequence from SBINIT is required before Active state can be re-established. The exit is also triggered by the local Adapter requesting Active on lp_state_req or by receipt of an Active Request sideband message from the remote partner, following which the RDI transitions to Reset and the NOP-to-Active request cycle is required before training begins.

Active → LinkReset and Active → Disabled. LinkReset is entered when the Adapter asserts lp_state_req equal to LinkReset or when a {LinkMgmt.RDI.LinkReset.Req} sideband message is received from the remote partner. Disabled is entered when the Adapter explicitly requests link shutdown through lp_state_req equal to Disabled. Both transitions require the stall handshake before the sideband negotiation begins. Disabled takes priority over LinkReset when both are requested simultaneously. From LinkReset, the RDI returns to Reset upon completion of the sideband handshake. Disabled is a terminal state; recovery requires a domain reset.

Active → LinkError. LinkError is entered immediately when lp_linkerror is asserted by the Adapter or when a fatal physical-layer fault is detected. Unlike all other Active-state exits, the transition to LinkError does not wait for the stall handshake; pl_state_sts may transition to LinkError before pl_rx_active_req de-asserts, though both pl_rx_active_req and lp_rx_active_sts must de-assert before pl_state_sts subsequently transitions away from LinkError. LinkError may be entered from any RDI state.

Stall Handshake. All graceful exits from Active — to Retrain, L1, L2, LinkReset, and Disabled — require the stall handshake to complete before any sideband negotiation begins. The Physical Layer asserts pl_stallreq to request that the Adapter flush all in-flight Flits and inhibit new Flit generation. The Adapter responds with lp_stallack when the data path is idle and aligned. The Physical Layer

must not initiate the state-change sideband message until lp_stallack is observed. This handshake ensures no Flits are orphaned during the transition.

Clock Request and Wake Handshakes. The pl_clk_req signal is used by the Physical Layer to prevent the Adapter from clock-gating its logic at critical points: before asserting pl_inband_pres, before state transitions that require the Adapter to observe level-sensitive status signals, and on PM exit when the Adapter must restore its clock activity. The Adapter must assert lp_clk_ack with at least one clock cycle of separation from pl_clk_req assertion before gating or ungating its clocks. The lp_wake_req signal provides a complementary path from the Adapter to the Physical Layer, used before lp_state_req changes to ensure the Physical Layer clocks are active. The Physical Layer acknowledges with pl_wake_ack.

Protocol and Configuration Advertisement. Following successful parameter negotiation during MBINIT.PARAM and protocol capability exchange during LINKINIT, the Physical Layer drives pl_protocol and pl_protocol_flitfmt to indicate the negotiated protocol stack and Flit format, with pl_protocol_vld asserted to indicate that these values are valid. The Adapter must sample and store these values when pl_protocol_vld is asserted and pl_state_sts is Reset with pl_inband_pres set. The signals pl_speedmode and pl_lnk_cfg convey the current operating data rate and active lane configuration respectively; these must be considered valid only when pl_state_sts is Active or Retrain. The pl_max_speedmode signal indicates whether the negotiated maximum data rate exceeds 32 GT/s and is valid only during the Reset-to-Active transition. The pl_phyinrecenter signal is asserted whenever the Physical Layer is engaged in training or retraining, informing the Adapter that the link is not yet carrying mission-mode data.

2.6 Register File Specification

The UCIe register architecture provides the software-visible configuration and status interface for the PHY and D2D Adapter logic. The specification defines two distinct address spaces: a Configuration Space accessed through PCIe Extended Capability configuration transactions, and a Memory-Mapped I/O Space accessed through standard memory read and write transactions. Software discovers the MMIO space location through pointer fields contained within the Configuration Space. Both spaces are accessible locally through the configuration bus and remotely through the sideband register access mechanism defined in Chapter 7 of the specification, subject to the constraint that remote sideband access is restricted to the two defined DVSEC capability structures and the four defined UCIe Register Blocks.

The Configuration Space is organized as a Designated Vendor Specific Extended Capability structure compliant with the PCIe Extended Capability framework. The MMIO space is partitioned into a D2D Adapter Register Block occupying the first 4 KB and a PHY Register Block occupying the second 4 KB, both discoverable through the Register Locator fields of the DVSEC structure. The PHY Register Block, which is the primary target of this implementation, contains mandatory registers beginning at offset 1000h, with extended capabilities beginning at offset 1200h.

2.6.1 Config Space Register Map

The Configuration Space of the UCIe Link DVSEC capability occupies a contiguous region beginning at the base address of the UCIe Register Block and contains nine register groups totalling 36 bytes in the minimal single-module, single-locator configuration implemented here. All register values are encoded in little-endian byte order.

PCI Express Extended Capability Header (Offset 000h). The Extended Capability Header identifies this structure to PCIe-aware software as a Designated Vendor Specific Extended Capability. The Capability ID field is fixed at 0023h, which is the standardized PCIe DVSEC capability identifier. The Capability Revision field is set to 1h. The Next Capability Offset field points to the next extended capability in the chain and is design-dependent; it is set to zero when no subsequent capability follows.

DVSEC Headers (Offsets 004h and 008h). The two DVSEC Header registers identify the UCIe consortium as the defining vendor and distinguish this structure from other vendor-specific capabilities. The Vendor ID field is fixed at D2DEh, the assigned identifier for the UCIe consortium. The DVSEC Revision field is 0h and the DVSEC Length field encodes the total byte length of the capability structure. The DVSEC ID field at offset 008h is 0000h for the UCIe Link DVSEC.

Capability Descriptor (Offset 00Ah). This register carries an implementation-defined descriptor encoding that software may use to identify the specific configuration of the UCIe implementation. Its interpretation is defined by the implementation and is outside the mandatory UCIe register specification.

UCIe Link Capability (Offset 00Ch). This register advertises the static hardware capabilities of the UCIe PHY implementation. It is initialized from hardware inputs at reset and thereafter treated as read-only. Table 2.x lists the defined capability bits relevant to this implementation.

UCIe Link Control (Offset 010h). This register provides the primary software interface for initiating link training and configuring optional features. It is fully software-writable subject to the auto-clear behaviors defined for specific bits. The Target Link Width field at bits [5:2] and the Target Link Speed field at bits [9:6] specify the desired operating configuration and are applied during the subsequent training sequence. The Start UCIe Link Training bit at bit [10] triggers a fresh link initialization when written to one; it is automatically cleared by hardware when training completes with either success or failure. The Retrain UCIe Link bit at bit [11] has the same auto-clear behavior and triggers a retraining sequence from the Active state. If software writes a one to either bit while training is already in progress, the write is ignored. Bits [21], [22], and [23] enable PMO, PSPT, and L2SPD respectively; a feature bit has effect only if the corresponding hardware capability bit in the Capability register is also set.

UCIe Link Status (Offset 014h). This register reflects the current operational state of the link and records error events. It contains a mix of three attribute types: read-only live status fields, RW1C fields that are set by hardware and cleared by software writing one, and RW1CS sticky fields that persist across soft resets and are cleared only by hardware reset or an explicit write-one-to-clear. Key status fields include the Link Training and Retraining Status bit, the Link Status bit indicating

whether the link is operationally up, the current negotiated link speed, and the current link width. Error-related fields in this register capture detected correctable and uncorrectable events and remain set until explicitly cleared by software.

Link Event Notification Control (Offset 018h) and Error Notification Control (Offset 01Ah). These two registers control the conditions under which the PHY generates an interrupt notification to the local software. Link Event Notification Control enables notification for link state change events such as transitions to and from Active, Retrain, or error states. Error Notification Control enables notification for correctable and uncorrectable error detections. Both registers are fully software-writable.

Register Locator 0 Low and High (Offsets 01Ch and 020h). The Register Locator pair forms a 64-bit pointer that software uses to locate the MMIO-mapped D2D/PHY Register Block. The low register carries a three-bit Base Address Register Indicator (BIR) identifying which BAR provides the base address for the MMIO region, a four-bit Register Block Identifier fixed at 0h to identify this locator as pointing to the D2D/PHY Register Block, and a 25-bit DWORD-granular offset from the BAR base to the start of the register block. The high register carries the upper 32 bits of the 64-bit offset for implementations requiring MMIO placement above the 4 GB boundary. Both registers are read-only. In this implementation, the Register Block Identifier is 0h, the BIR points to BAR 0, and the offset is fixed by the implementation's MMIO address allocation.

2.6.2 MMIO PHY Register Block

The D2D/PHY Register Block occupies 8 KB of memory-mapped address space. The first 4 KB of this block, beginning at offset 0h within the block base, is allocated to the D2D Adapter. The second 4 KB, beginning at offset 1000h, contains the Physical Layer register set that is the subject of this section. Access to the PHY MMIO block is performed through Memory Read and Memory Write sideband transactions using the MMIO address space opcode encodings. All accesses where the Register Locator field of the address is non-zero are rejected with an Unsupported Request completion. Accesses to offsets outside the defined register windows are similarly rejected.

The PHY Register Block contains thirteen registers or register pairs, organized into four functional groups: static capability advertisement, dynamic control and status, training calibration setup, and error capture and diagnostic logging. Table 2.x lists the complete PHY MMIO register map for this implementation.

of the Physical Layer. It is initialized at reset from hardware inputs reflecting the implemented configuration and is thereafter treated as read-only. Defined capability fields include terminated link support, transmitter equalization support, the set of supported transmitter voltage swing encodings, the receiver clock mode options available for operating speeds at or below 32 GT/s, the supported clock phase configuration, the package type indicator distinguishing Standard Package from Advanced Package, the tightly coupled mode support bit, and the Transmitter Adjustment for Runtime Recalibration capability. Because TARR is excluded from this implementation's feature scope, the corresponding capability bit is hardcoded to zero.

PHY Control (Offset 1004h). The PHY Control register provides software-accessible overrides for analog front-end operating parameters. Defined fields include the receiver termination enable, the transmitter equalization enable, the receiver clock mode selection between strobe and free-running modes for operating speeds at or below 32 GT/s, the receiver clock phase selection between differential and quadrature, a force-x8-width-mode bit that configures an x16 Standard Package module to operate in the x8 width configuration under software direction, and fields for forcing specific I/Q correction and transmitter equalization preset values. These control bits override the values determined autonomously by the training sequence when asserted and revert to training-determined values when de-asserted.

PHY Status (Offset 1008h). The PHY Status register presents live read-only visibility into the current operating state of the analog front-end. Fields reflect the active state of receiver termination, the active state of transmitter equalization, the current clock mode in use, the current clock phase configuration, the lane reversal status indicating whether lane reversal was applied during the most recent training, the active I/Q correction parameter value, the active transmitter equalization preset setting, and the TARR status.

PHY Initialization and Debug (Offset 100Ch). This register combines software-writable initialization control fields with read-only diagnostic status fields. The control fields provide software-accessible hooks for influencing the initialization sequence behavior and are consumed by the LTSM during bring-up. The read-only fields provide runtime diagnostic visibility. Because these fields serve implementation-defined initialization and debug purposes beyond the mandatory register definitions, their complete field allocation is described in Chapter 6.

Training Setup Registers (Offsets 1010h, 1020h, 1030h, 1050h). The four Training Setup registers carry the software-configurable parameters that govern the scope and duration of the MBTRAIN calibration substates. These parameters include the phase interpolator sweep range and step size, the voltage reference sweep parameters, the deskew calibration range, and the iteration counts used during the point test measurement windows. The register file exposes these values to the LTSM at the start of each training session; the LTSM distributes them to the active substate logic without modifying them. The complete field definitions for each training setup register are presented alongside the architecture description of the corresponding training substates in Chapter 6.

Current Lane Map Module 0 (Offset 1060h). This 64-bit register records the lane map configuration agreed upon by the two link partners following width degradation resolution during MBINIT.REPAIRMB or MBTRAIN.REPAIR. The lower half records the transmit lane map and the upper half records the receive lane map, using the three-bit lane map encoding defined in Section 2.2.4. The LTSM writes this register at the conclusion of each successful repair negotiation. Software may read this register to determine the effective operating width of the link following any degradation events.

Error Log 0 (Offset 1080h). This register captures a snapshot of the link state at the moment a TRAINERROR condition is detected. All fields are Read-Only Sticky: they are written once by hardware on the first error event following reset and are not overwritten by subsequent events

unless the register is cleared by a hardware reset. The lower byte captures the LTSM state code identifying the specific substate in which the error occurred, using the eight-bit substate encoding defined by the specification's error logging table. Additional fields capture whether lane reversal was in effect at the time of the error and whether a width degradation had been applied. This register provides the primary diagnostic entry point for software-initiated link failure analysis.

Error Log 1 (Offset 1090h). This register captures categorical error event indicators using a mix of Read-Only Sticky and Write-One-to-Clear Sticky attributes. RW1CS fields are set by hardware when specific error conditions occur and persist across soft resets, requiring an explicit software write-one-to-clear operation to de-assert them. Captured error categories include LTSM watchdog timeout, sideband message exchange timeout, remote link partner error detection, and internal PHY error events. The sticky behavior ensures that transient error events that resolve before software intervention remain visible for post-error diagnostic inspection.

Runtime Link Test Control (Offset 1100h). This 64-bit register configures the runtime link testing feature, which supports pattern-based link quality assessment during the ACTIVE state and during PHYRETRAIN. Control fields include the enable for initiating a runtime test, the Apply Lane Repair bit that signals the PHYRETRAIN state machine to evaluate width degradation during the subsequent retrain session, and parameters governing the test pattern and measurement interval. The detailed field definitions are described in Chapter 6 alongside the runtime link testing architecture.

Runtime Link Test Status (Offset 1108h). This register provides live read-only visibility into the runtime link test execution state. The primary field is the busy flag, which is asserted by hardware while a runtime link test is in progress and de-asserted when the test completes or is aborted. The PHYRETRAIN state machine reads this field to determine whether a runtime link test was active at the time the retrain was triggered, which influences the retrain encoding selection as described in Section 2.4.4.

2.6.3 Register Attribute Enforcement

Attribute	Description
RO	Read Only
ROS	Read Only Sticky[a]
RW	Read-Write
RWL	Read Write Lock Follow RW behavior until locked. When locked, the bit value cannot be altered by software. The locking condition associated with each RWL field is specified as part of the field definition.
RWO	Read-Write-One-To-Lock Field becomes RO after writing 1 to it. Cleared by management reset.
RWS	Read Write Sticky[a]
RW1C	Read-Write-One-To-Clear
RW1CS	Read-Write-One-To-Clear-Sticky[a]
HWInit	Hardware Initialized[b]
RsvdP	Reserved and Preserved
RsvdZ	Reserved and Zero

Table 2-10: Register attributes

The UCIe specification defines a set of register field attributes that govern how each bit field responds to hardware events and software access operations. These attributes are defined in Table 9-4 of the specification and apply uniformly across all registers in both the Configuration Space and the MMIO PHY Register Block. Their consistent enforcement is a mandatory compliance requirement: a register implementation that violates any attribute's behavioral rule is not specification-conformant. This section defines each attribute, the enforcement rule it implies, and the conditions under which each rule applies.

RO Enforcement. Fields with the RO attribute may be sourced either from a fixed hardwired constant or from a live hardware signal whose value reflects the current operating state of the PHY. In either case, software writes targeting an RO field are accepted by the access protocol — they generate a Successful Completion response — but the write data is discarded and the field value remains unchanged. This behavior distinguishes RO from an access error: the specification does not require a UR or CA completion for writes to RO fields.

HWInit Enforcement. HWInit fields are sampled from hardware input signals during domain reset and latched into read-only storage at the conclusion of reset. They are not re-sampled during soft resets or retraining events. This attribute is used for capability fields whose values may vary between silicon integration contexts — for example, the maximum link speed field may be configured by a package-level strap or firmware initialization to reflect the physical routing constraints of a given assembly, even if the silicon supports higher speeds. Once the domain reset sequence is complete, HWInit fields are indistinguishable from RO fields to software.

RW Enforcement. Software writes to RW fields update the stored register value, which is subsequently presented to the hardware logic that consumes it. The new value takes effect at the first clock edge after the write transaction completes. Reads of RW fields return the last value written by software; hardware logic does not modify an RW field's stored value. Byte enable signals associated with the write transaction determine which bytes of the register are updated; bytes whose enable is deasserted retain their previous value.

RW1C and RW1CS Enforcement. These attributes support event latching where hardware must signal that a condition occurred and software must explicitly acknowledge it. Hardware sets the bit by asserting it to one on the triggering event; this action takes priority over a simultaneous software clear, ensuring that no event is silently lost. Software clears the bit by writing a one to the field position in a write transaction; writing zero has no effect. The distinction between RW1C and RW1CS is the scope of reset that clears the field: RW1C fields are cleared by any reset including soft resets and domain resets, while RW1CS fields are cleared only by a domain reset or by the software write-one-to-clear action. In the UCIe Link Status register, error detection fields use the RW1CS attribute specifically to ensure that error events that occurred during a training session remain visible to software after a retraining cycle that involves a soft reset.

ROS Enforcement. ROS fields capture a snapshot of hardware state at the moment of the first qualifying error event following a domain reset. If subsequent error events occur before software reads and clears the field, the original captured value is preserved and the new event is not logged in the same location. This first-event capture behavior ensures that the root-cause state is available

for software analysis even when error events recur. Only a domain reset returns an ROS field to its default value; soft resets and retraining do not affect it. Software writes to ROS fields are silently ignored.

RsvdZ and RsvdP Enforcement. RsvdZ fields always return zero on reads regardless of any value previously written. Software must not assume that a zero read from a RsvdZ field means the field is available for use; these fields are reserved for future specification extensions. RsvdP fields require that software perform a read-modify-write pattern when writing to a register containing them: the software reads the current register value, modifies only the non-reserved fields, and writes back the complete value with the RsvdP fields restored to their read values. Failure to preserve RsvdP fields may cause undefined behavior in future-compatible implementations that assign meaning to those bit positions.

Auto-Clear Behavior. The Start UCIe Link Training bit (UCIe Link Control offset 010h, bit [10]) and the Retrain UCIe Link bit (bit [11]) have a specialized auto-clear behavior that overlays the standard RW attribute. When software writes a one to either bit, the hardware initiates the corresponding training or retraining sequence and holds the bit set while the sequence is in progress. The hardware automatically clears the bit to zero when the training sequence concludes with either a success or a failure outcome. Software may poll the bit to determine when the training action has completed. If software writes a one to either bit while training is already active, as indicated by the Link Training and Retraining Status bit in the UCIe Link Status register, the write is ignored without affecting the ongoing training sequence.

Mixed-Attribute Register Enforcement. Several registers in the UCIe register space contain fields of different attribute types within the same 32-bit or 64-bit register word. The UCIe Link Status register, for example, combines RO live-status fields with RW1C event-latch fields and RW1CS sticky error fields. Enforcement is applied per bit position: a single write transaction may simultaneously attempt to clear an RW1C field in one bit position while the RO fields in other positions remain unaffected. The enforcement logic must correctly apply the attribute rule for each bit independently, regardless of the mixed nature of the containing register.

Address Error and Unsupported Request Conditions. Any sideband register access whose decoded address falls outside the defined register windows, or whose Register Locator field is non-zero, is rejected with an Unsupported Request completion status. The UR response is returned via the sideband completion mechanism with a status field encoding of 001b. No register state is modified by an access that generates a UR response. This behavior ensures that software can use UR completion status as a reliable indicator that the addressed register does not exist in the current implementation, enabling capability discovery through probing.

Reset Scope and Sticky Field Persistence. The specification distinguishes between domain resets and soft resets in the context of sticky field persistence. A domain reset initializes all register fields, including ROS and RW1CS fields, to their specified default values. Soft resets — which may occur during LTSM transitions, sideband initialization sequences, or retraining events that do not require a full hardware reset — do not affect the stored values of sticky fields. This distinction ensures that

error information captured in the Error Log registers during a failed training attempt remains available for software diagnostic purposes across the retraining cycles that follow the failure.

3.0 System Architecture

Chapter 2 established the specification requirements governing the UCIe Standard Package PHY — the module signal structure, mainband and sideband behavioral rules, the LTSM and RDI state machines, and the register organization defined by UCIe Specification Revision 3.0. This chapter translates those requirements into a concrete implementation architecture. It describes how the digital PHY is partitioned into subsystems, what boundary was drawn between the digital logic and the abstracted analog front-end, how the clock domains are structured, how the subsystems communicate across those clock domains, and what parameters govern the top-level configuration. The architectural decisions described here form the structural foundation upon which the subsystem-level designs presented in Chapters 4 through 7 are built.

3.1 Design Philosophy and Methodology

The overarching principle governing the architectural approach taken in this implementation is strict specification traceability. Every structural decision — how the design is partitioned, where domain boundaries are placed, which blocks are grouped together, and which interfaces are defined between them — is grounded in a corresponding requirement or behavioral rule drawn from UCIe Specification Revision 3.0. No major architectural decision is made solely for implementation convenience without a specification anchor; where implementation choices are made within the latitude the specification permits, those choices are explicitly identified as such in the sections that follow.

A second principle is the separation of functional concerns. The UCIe specification itself establishes a layered decomposition: sideband messaging, mainband data transport, link training sequencing, adapter-facing state abstraction, and software-visible configuration are each governed by distinct sections with distinct behavioral rules, timing relationships, and clock domain requirements. The implementation reflects this decomposition directly: rather than consolidating unrelated functions into monolithic blocks for the sake of simplicity, each major functional domain is mapped to a discrete subsystem with well-defined ports. This makes each subsystem independently verifiable and independently readable, which is consistent with the academic reference objective stated in the project contributions.

A third principle is the minimization of implicit dependencies. Where two subsystems must exchange information, that exchange is made explicit through a defined inter-subsystem interface carrying named signals with documented directionality and clock domain ownership. The register file, for example, does not directly drive mainband control logic; it exposes configuration outputs that the main state machine reads and distributes. The sideband subsystem does not directly manipulate LTSM state; it passes received messages to the main state machine through an asynchronous FIFO-based crossing. These explicit boundaries preserve the subsystem isolation that makes independent subsystem verification feasible.

3.1.1 Subsystem Decomposition Strategy

The implemented PHY is partitioned into four top-level subsystems, each corresponding to a coherent and self-contained functional domain of the UCIe specification: the Mainband datapath (unit_mb_die), the Sideband subsystem (SideBand_Top), the Main State Machine (MainSM), and the Register File (Reg_File). The UCIe_PHY module serves as the system integration wrapper, instantiating these four subsystems and connecting them through the inter-subsystem interfaces described in Section 3.4. No logic beyond signal routing and port connection resides at the UCIe_PHY level; all behavioral implementation is contained within the subsystems themselves.

The Mainband datapath subsystem encapsulates all logic associated with the high-speed parallel data path: byte-to-lane mapping, scrambling and descrambling, training pattern generation and comparison, lane reversal detection and application, width degradation, and the serialization and deserialization interfaces to the analog front-end. It is a purely datapath-oriented block; it does not control its own operational mode. Instead, it receives mode-select and enable signals from the Main State Machine and presents training result signals back to it. This separation ensures that the training sequencing logic, which is tightly specified by the LTSM state machine, remains entirely within the MainSM subsystem rather than being distributed across the datapath.

The Sideband subsystem encapsulates the complete sideband protocol engine. It operates on a permanently powered auxiliary clock domain and is therefore structurally independent of the mainband clock infrastructure. It handles packet serialization and deserialization, inbound and outbound message routing, register access transaction processing, RDI message forwarding to the Main State Machine, and the negotiation and enforcement of Sideband Feature Extensions and Sideband Performant Mode Operation. Because the sideband channel must remain operational before the mainband is trained and during low-power states in which the mainband clock is absent, its self-contained clock domain and independent power behavior make it the natural candidate for a discrete top-level subsystem.

The Main State Machine subsystem groups the LTSM controller and the RDI state machine into a single wrapper. This grouping reflects two facts from the specification. First, both machines operate on the same mainband functional clock domain (lclk) and must exchange state information and handshaking signals synchronously within that domain. Second, the RDI state machine is not an independent actor: its transitions are tightly coupled to the LTSM progress, and several LTSM state entry and exit conditions depend on the current RDI state. Encapsulating both within a single subsystem allows their handshaking to be implemented as direct intra-subsystem signal connections rather than inter-subsystem interfaces, avoiding the clock domain crossing complexity that would arise if they were partitioned at the top level.

The Register File subsystem is accessed via the sideband clock domain, as all register read and write transactions are delivered to it by the sideband register access engine over a dedicated bus. It exposes configuration outputs that are consumed by the Main State Machine and the Mainband datapath, and it captures hardware-driven status and error signals from both. Its position as a discrete subsystem with a well-defined access bus reflects the specification's requirement that all software-visible configuration and status be mediated through a formally specified register map,

and that access attribute enforcement — distinguishing read-only, read-write, hardware-initialized, and write-one-to-clear fields — be consistently applied.

3.1.2 Analog Abstraction Boundary Definition

The UCIe specification defines the physical interface of a Standard Package Module in terms of external observable behavior — the signals present at the bump level, their electrical characteristics, and the training procedures required to bring the link into operation — without mandating how the PHY is partitioned between analog and digital circuitry. The analog abstraction boundary is therefore an implementation decision. This section defines where that boundary is drawn in this design and what it implies for the structural organization of the four subsystems.

The boundary separates two functionally distinct categories of logic. On the analog side reside all functions that operate directly on continuous-time or high-speed serial physical signals: clock generation and low-level distribution, high-speed serialization of outbound parallel data, deserialization of inbound serial streams sampled on the forwarded receive clock, and the tri-state output drivers that present or suppress signal activity on the physical bumps. These functions are inherently dependent on process, voltage, and temperature characteristics and are not amenable to synchronous digital modeling. On the digital side reside all protocol-level functions — byte-to-lane mapping, scrambling, training pattern generation and comparison, lane reversal, width degradation, valid framing, clock pattern generation, clock activity detection, state machine operation, sideband protocol processing, and register management — which are fully expressible as synchronous logic operating on parallel-domain clocks derived from the analog clock hierarchy.

The signals that cross this boundary fall into three groups. The first is clock delivery: the analog front-end generates and distributes the mainband parallel-domain clock (lclk), a gated variant of it (gated_lclk), the raw PLL output (pll_clk), and its measured period (pll_period), as well as the sideband functional clock (clk_sb) derived from an independent sideband PLL and divider chain. These clocks are consumed by the digital subsystems as synchronous references; no clock is generated within the digital domain itself.

The second group comprises outward control signals through which the digital logic programs the operating state of the analog front-end. For the mainband, this includes the PLL speed selection, transmit and receive path enable signals, tri-state lane select codes for the clock, track, valid, and data signal groups, and the voltage comparator enable and completion handshake. For the sideband, it includes the serializer pattern mode select and the PMO enable that governs whether the high-speed serial mode is engaged. Calibration codes computed during the MBTRAIN sub-states — phase interpolator settings and voltage reference DAC values — are driven outward to the analog front-end at the appropriate points in the training sequence; the detailed handling of these is addressed in Chapter 4.

The third group comprises inward status and result signals through which the analog front-end returns observable outcomes to the digital subsystems. For the mainband, this is the recovered parallel data words assembled from the received serial stream, the data valid qualifier, and the received valid framing data and framing qualifier. For the sideband, it is the deserialized receive

data and its valid qualifier. These are the only signals the digital logic observes from the physical channel; all link training decisions and error assessments are made exclusively from this parallel-domain representation of the received signal.

This boundary definition ensures that the digital implementation is fully independent of the specific analog IP technology used in integration. Substituting a different analog front-end — provided it respects the same parallel-domain interface contract — requires no modification to any of the four digital subsystems. This independence motivates documenting the analog abstraction boundary as a first-class architectural element rather than treating it as an implementation detail.

3.2 Top-Level Module Hierarchy

The physical hierarchy of the implemented design consists of a single integration wrapper, UCIe_PHY, beneath which the four subsystems — unit_mb_die, SideBand_Top, MainSM, and Reg_File — are instantiated as peer blocks at the same hierarchical level. This section describes the role and port structure of the UCIe_PHY wrapper, the responsibilities of each subsystem, and the directional data flow that connects them during link initialization, training, and active operation.

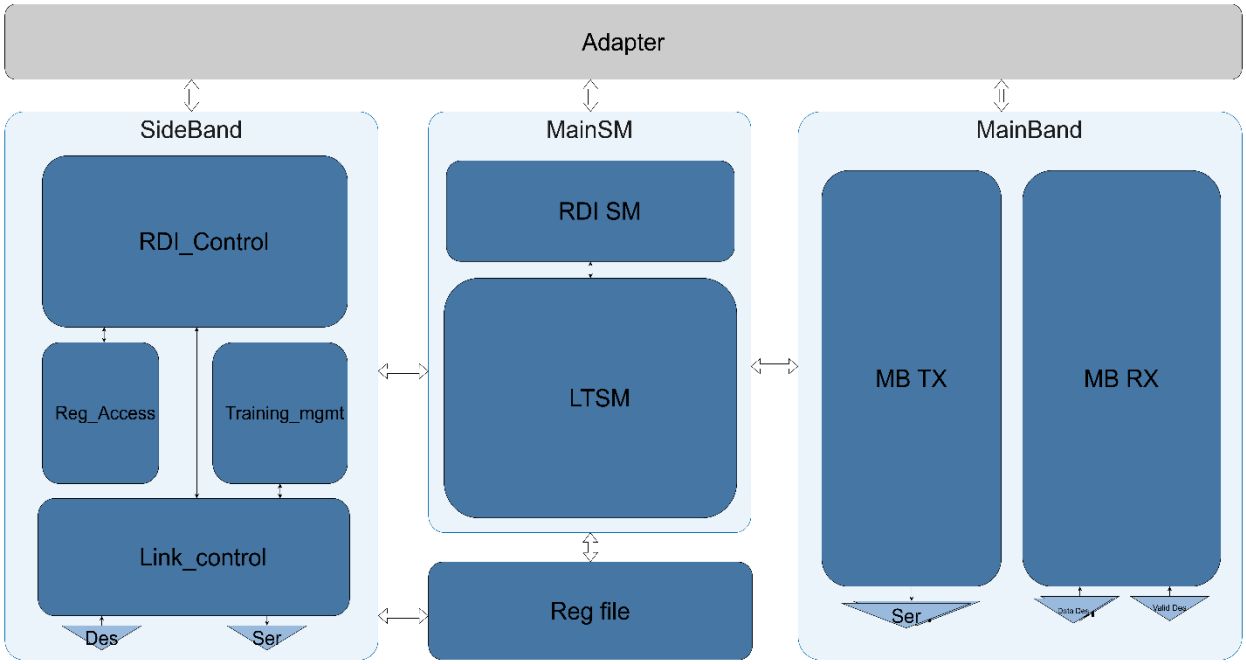


Figure 3-1: Top Level Module Hierarchy

3.2.1 UCIe PHY as System Integration Wrapper

UCIe_PHY is a structurally pure integration module. It contains no behavioral logic, including registers, state machines, or combinational control logic. Its sole function is to define the complete external port boundary of the implemented PHY, instantiate the four subsystems, and connect them to each other and to the external ports through named nets. All design behavior is contained within the subsystems themselves.

The external ports of UCIe_PHY fall into three categories. The first is the analog front-end boundary, which represents the physical channel side of the design as defined by the analog abstraction boundary described in Section 3.1.2. On the mainband side, this comprises the per-lane serial receive data pins, the received valid, clock, and track pins, and the corresponding transmit counterparts. On the sideband side, it comprises the received serial clock and data pins and the transmitted serial clock and data pins. These ports connect directly into unit_mb_die and SideBand_Top respectively; no additional processing or buffering is introduced at the wrapper level.

The second category is the RDI adapter-facing interface, which is the externally visible contract between this PHY and the D2D Adapter above it, as specified by UCIe. This interface carries three classes of signals: state coordination signals, through which the Adapter requests and the PHY reports link power and operational states; the mainband parallel data path, through which the Adapter presents outbound data and the PHY delivers inbound recovered data; and the sideband configuration channel, through which configuration transactions are exchanged using the pl_cfg and lp_cfg signal groups. All RDI state coordination signals are routed to and from MainSM.

The third category is the set of top-level static capability straps and configurable parameters. These are design-time inputs that establish the PHY's advertised capabilities — maximum link width, maximum link speed, packaging type, supported clock modes, and feature support flags — and are driven into Reg_File to populate the hardware-initialized fields of the capability registers. They are not driven by any subsystem; they represent configuration choices fixed at integration time, described further in Section 3.5.

Clock ownership follows subsystem functionality: clk_sb is generated within SideBand_Top, which owns the sideband clock generation logic. The UCIe_PHY wrapper simply forwards this clock to Reg_File, the only other subsystem operating in the clk_sb domain.

3.2.2 Four Top-Level Subsystems

MainBand is the mainband datapath subsystem. It encompasses the complete transmit and receive paths for the high-speed parallel data channel: byte-to-lane mapping, LFSR-based scrambling and descrambling, training pattern generation and comparison for per-lane error detection, lane reversal detection and correction, width degradation logic, valid framing generation and detection, clock pattern generation for training, and clock activity detection on the received forwarded clock. The subsystem is purely a datapath consumer of control: it does not determine its own operational mode. All mode-select, enable, and training-phase control signals are received from MainSM, and all training results — pattern completion indicators, per-lane error counts, lane reversal detection outcomes, and valid frame detection status — are reported back to MainSM. On its upper boundary, unit_mb_die exchanges the mainband parallel data words with the D2D Adapter via the RDI data path. On its lower boundary, it presents pre-serialization parallel data to the analog front-end and receives recovered parallel data from it.

SideBand_Top is the sideband subsystem. It operates entirely in the clk_sb domain derived from its internal PLL and divider chain, making it independent of the mainband clock and operational

under all link power states. The subsystem handles sideband packet serialization and deserialization, inbound packet routing, outbound packet arbitration, register access transaction processing, SFES negotiation and enforcement, and PMO-conditioned serial mode selection. It maintains two message paths to MainSM: an LTSM message path, which carries training-phase sideband protocol messages to and from the LTSM controller, and an RDI message path, which carries link management messages to and from the RDI state machine. Both paths cross the clk_sb-to-lclk domain boundary through asynchronous FIFOs internal to SideBand_Top. On its upper boundary, SideBand_Top exchanges sideband configuration transactions with the D2D Adapter over the pl_cfg and lp_cfg signal groups. On its lower boundary, it presents and receives the serial sideband bit stream to and from the analog front-end.

MainSM is the main state machine subsystem, wrapping the LTSM controller and the RDI state machine in a single lclk-domain block. The LTSM governs the complete link lifecycle across its eight top-level states and thirteen MBTRAIN sub-states, issuing all control signals to unit_mb_die and dispatching all outbound sideband training messages through SideBand_Top. The RDI state machine runs as a coordinated peer of the LTSM, managing the power and operational state handshakes on the adapter-facing boundary and reporting link state to the Adapter through pl_state_sts. The two machines share direct intra-subsystem signal connections for state coordination; their handshaking does not cross a clock domain boundary. On its external boundaries, MainSM interfaces upward to the Adapter via the full set of RDI state coordination signals, sideways to SideBand_Top via the two message paths and the SBINIT pattern control interface, sideways to unit_mb_die via the training control and result buses, and downward to Reg_File for configuration consumption and hardware status reporting.

Reg_File is the register file subsystem. It implements both the UCIe Link DVSEC register block and the PHY-specific MMIO register space, enforcing per-field access attributes across all register types. It operates entirely in the clk_sb domain and is accessed exclusively through the register access bus driven by SideBand_Top. On its output side, it distributes software-programmed control values — target link width, link speed, feature enables, training start, and lane mask — to MainSM. On its input side, it captures hardware-driven status from MainSM, including negotiated link width and speed, lane reversal status, LTSM state history, and training error information recorded in the error log registers.

3.2.3 Architectural Data Flow

The flow of data and control through the four subsystems differs across the three principal phases of PHY operation: link initialization and training, active data transfer, and register access.

During link initialization and training, the controlling actor is MainSM. The LTSM sequentially enables the training phases in unit_mb_die by asserting mode-select and enable signals appropriate to each LTSM sub-state, and monitors the completion and result signals that unit_mb_die returns. Concurrently, it dispatches outbound sideband training messages to SideBand_Top over the LTSM message path and receives the corresponding partner responses. The coordination of local training progress with the remote die's progress is thus carried through SideBand_Top on the sideband channel, while the physical training measurements are made by

unit_mb_die on the mainband channel. As training sub-states complete, MainSM updates its status outputs, which the Reg_File captures to maintain a current hardware-driven view of link state accessible to software.

During active data transfer, MainSM asserts the mapper enable signal to unit_mb_die, authorizing the datapath to pass traffic. Outbound data presented by the Adapter on the RDI data path enters unit_mb_die, which maps, scrambles, and presents it to the analog front-end for serialization. In the receive direction, the analog front-end delivers recovered parallel data to unit_mb_die, which descrambles and demaps it and presents it to the Adapter on the RDI data path. MainSM continues to monitor the datapath for error conditions and responds to any Adapter-initiated stall or error via the RDI state coordination signals. SideBand_Top remains active throughout this phase, forwarding any link management messages arriving on the sideband channel to the RDI state machine in MainSM.

Register access transactions follow a dedicated path that is independent of both the training and data flows. An inbound register access packet arriving on the sideband channel is received by SideBand_Top, depacketized by its internal register access engine, and presented to Reg_File over the register access bus as an addressed read or write transaction. Reg_File executes the transaction, enforces the applicable access attribute, and returns the completion data to SideBand_Top, which assembles and forward to the local adapter. This path operates entirely in the clk_sb domain and is available throughout all LTSM states, including RESET, as the sideband subsystem and register file share the always-on power domain.

3.3 Clock Domain Architecture

The implemented PHY operates across two independent clock domains, each derived from a separate PLL and each consisting of two frequency tiers: a high-speed serialization clock driving serial-domain logic, and a lower-frequency parallel-domain clock driving all protocol-level digital logic. The mainband domain is sourced from the mainband analog PLL and serves unit_mb_die and MainSM. The sideband domain is sourced from an independent sideband PLL internal to SideBand_Top and serves SideBand_Top and Reg_File. The two domains share no common clock source and have no phase relationship; all signal exchanges between them are handled through asynchronous FIFO-based clock domain crossings, which are described at the subsystem level in Chapters 5 and 6.

The sections that follow describe each of the four clock signals in the design — two per domain — covering their source, frequency relationship, distribution, and the logic they clock.

3.3.1 High-Speed MainBand Serialization Clock

The mainband high-speed serialization clock, pll_clk, is generated by the mainband analog PLL and delivered to the digital subsystem as an input across the analog abstraction boundary. It is the highest-frequency clock in the design and the timing reference for all serial-domain operations on the mainband channel.

On the transmit side, pll_clk drives the shift registers of the data lane serializers (Data_ser) and the valid lane serializer (Valid_ser), clocking out one bit per edge on each serial output. Serialization operates in double data rate (DDR) mode, producing two bits per pll_clk cycle per lane, such that a 32-bit parallel lane word presented from the parallel domain is fully serialized in 16 pll_clk cycles. This DDR ratio of 16 establishes the relationship between pll_clk and the mainband parallel-domain clock: pll_clk ÷ 16 yields the frequency of the parallel reference clock from which the mainband logic clock is derived.

On the receive side, the data lane deserializers use pll_clk as the DDR sampling clock, capturing two bits per cycle from the incoming serial stream. The clock pattern detector (CLK_PATTERN_DETECTOR_RX) also operates on pll_clk, since the forwarded clock and track patterns received from the remote die toggle at the pll_clk rate; sampling them at the parallel-domain clock frequency would alias the pattern and prevent detection. The transmit clock pattern generator (clk_pattern_gen_TX) similarly uses pll_clk to synthesize and drive the forwarded differential clock output and track lane during the clock training sub-state.

pll_clk is never gated by the digital subsystem. It runs freely from the analog PLL as long as the PLL is locked, independent of the LTSM state or the mainband traffic state. The measured period of pll_clk (pll_period) is forwarded as a real-valued parameter from the analog front-end to the digital subsystem and is used by the LTSM training sub-states for timing reference calculations. The digital domain exercises no control over pll_clk's generation; all PLL management is performed through the speed selection and enable signals driven outward across the analog abstraction boundary, as described in Section 3.1.2.

Table 3-1: Forward Clock Frequency and phase

Data rate (GT/s)	Clock freq. (fCK) (GHz)	Phase-1	Phase-2	Deskew (Req/Opt)
64	16	45	135	Required
48	12	45	135	Required
32	16	90	270	Required
	8	45	135	Required
24	12	90	270	Required
	6	45	135	Required
16	8	90	270	Required
12	6	90	270	Required
8	4	90	270	Optional
4	2	90	270	Optional

3.3.2 MainBand & MainSM Functional Clock

The mainband functional clock is the parallel-domain clock of the mainband subsystem and the Main State Machine. It is derived from pll_clk by a divide-by-16 stage in the analog front-end, producing div_clk, which at the DDR serialization ratio of 16 represents the rate at which one complete 32-bit parallel lane word is exchanged per cycle. div_clk is then passed through an

integrated clock gating (ICG) cell — clk_gate — whose enable input is driven by the LTSM. This gating stage produces the gated version of the clock, referred to as gated_lclk (or equivalently mb_clk within unit_mb_die), which drives all mainband parallel-domain protocol logic: Mapper, LFSR_TX, Valid_TX, the parallel-load sides of Data_ser and Valid_ser, and all parallel-domain receive blocks including the descrambler, demapper, and pattern comparator. It is also the clock presented to LTSM_TOP within MainSM.

The ungated variant, lclk, is distributed in parallel with gated_lclk and represents div_clk without the ICG stage. lclk is used by the RDI_SM and certain integration-level flops within MainSM that must remain clocked regardless of the LTSM's gating state, specifically to handle RDI power management handshakes and wake requests that can arrive while the LTSM has suppressed the gated clock. The two-clock arrangement within MainSM — lclk for the RDI_SM and gated_lclk for the LTSM — therefore reflects a specification-driven requirement: the adapter-facing state machine must be independently responsive to adapter transitions, including those that occur during low-power states in which the training state machine is deliberately frozen.

Clock gating is controlled by unit_gating_logic within the RDI_SM, which evaluates the current pl_state_sts, the inband_pres indicator, the phy_start signal, and the RDI stall and wake handshake state to determine when gating is permissible. The UCIe specification permits coarse clock gating in the RESET, LinkReset, Disabled, and power management states. When the gating condition is met, gated_lclk is held low, eliminating dynamic power in all mainband parallel-domain logic. The ICG cell ensures that the resulting clock edge is glitch-free, preventing spurious register transitions at the moment of gating or ungating.

gated_lclk is also forwarded from unit_mb_die into SideBand_Top as the write-domain clock for the asynchronous FIFOs in Training_Mgmt, where outbound LTSM sideband messages are enqueued on the lclk side and dequeued on the clk_sb side. This distribution is the single point at which the mainband and sideband clock domains interact structurally; the FIFOs internal to Training_Mgmt handle the metastability isolation.

3.3.3 High-Speed SideBand Serialization Clock

The sideband high-speed serialization clock is generated by the sideband PLL (sb_pll) inside SideBand_Top, entirely independently of the mainband analog PLL. The UCIe specification mandates that the sideband serial-domain clock be fixed at 800 MHz, irrespective of the mainband data rate or link state; sb_pll operates at this frequency to provide the serialization reference, but 800 MHz is not the operating frequency of the digital sideband logic — the parallel-domain clock at which all protocol processing runs is 100 MHz, produced by the divide-by-eight ClkDiv stage that follows sb_pll and is described in Section 3.3.4.

This clock, referred to internally as sb_pll_clock, drives the serial-domain logic of the sideband serializer and deserializer. In the transmit direction, it clocks the shift register of sb_serializer, shifting one bit per cycle onto the TXDATASB output pin at 800 Mbit/s in single data rate (SDR) mode. The forwarded sideband clock output, TXCKSB, is derived from sb_pll_clock through a gating

cell internal to the serializer that suppresses the clock output when no transmission is in progress, as required by the sideband protocol. In the receive direction, the deserializer samples incoming RXDATASB data using the received forwarded clock RXCKSB, which is the remote die's equivalent of TXCKSB at the same nominal frequency.

The serialization ratio between sb_pll_clock and the sideband parallel-domain clock is eight, consistent with the divide-by-8 stage that produces clk_sb: a 64-bit sideband packet is serialized over 64 sb_pll_clock cycles in standard mode, or equivalently over 8 clk_sb cycles. sb_pll_clock is not gated and is not distributed outside the sideband subsystem; it is an internal clock consumed exclusively by the sideband serial-domain hardware.

3.3.4 SideBand Functional Clock

clk_sb is the parallel-domain clock of the sideband subsystem and the register file. It is derived from sb_pll_clock by the divide-by-8 ClkDiv stage inside SideBand_Top, producing a 100 MHz clock, consistent with the auxiliary clock frequency recommended by the UCIe specification for the always-on sideband power domain.

clk_sb drives every digital parallel-domain block within SideBand_Top: the Link_Controller, the Training_Mgmt receive-side and transmit-arbitration logic, the RDI_control downstream and upstream paths, and the Reg_Access engine. It is forwarded from SideBand_Top to Reg_File through the UCIe_PHY wrapper, making it the sole clock of the register file subsystem. Because clk_sb originates within the sideband subsystem and Reg_File has no mainband clock input, all register accesses — regardless of whether they originate from link training activity or from adapter-initiated transactions during active operation — occur in the clk_sb domain.

clk_sb is never gated. The always-on character of the sideband power domain requires that sideband processing remain functional across all mainband power states, including states in which the mainband PLL is disabled or the gated_lclk is suppressed. Holding clk_sb permanently active ensures that sideband packets can be received and processed, register accesses can be serviced, and RDI link management messages can be forwarded to the RDI_SM at any point in the link lifecycle. The asynchronous FIFOs in Training_Mgmt bridge the clk_sb domain to the lclk domain at the single point where the sideband and mainband clock domains must exchange data, preserving the independence of the two domains at all other points in the architecture.

3.4 Inter-Subsystem Interfaces

The four subsystems communicate through four explicitly defined inter-subsystem interfaces. Each interface carries signals in a single defined direction or as a named bidirectional pair, operates in a stated clock domain, and has no implicit shared state with other interfaces. This section describes each interface in terms of its functional purpose, the signals it comprises, its clock domain ownership, and any domain crossing mechanism it employs.

3.4.1 MainSM ↔ Sideband

The interface between MainSM and SideBand_Top is the most structurally complex of the four, as it spans the lclk-to-clk_sb clock domain boundary and carries three functionally distinct signal groups.The first is the LTSM message path, which conveys training-phase sideband protocol messages in both directions across the clock domain boundary. In the transmit direction, the LTSM asserts a message identifier, MsgInfo field, data field, and a valid pulse on the lclk-domain side of Training_Mgmt's internal asynchronous FIFO; the clk_sb-domain side dequeues and serializes the message for transmission. In the receive direction, a received and decoded sideband message is enqueued on the clk_sb side and presented to the LTSM on the lclk side as a valid-qualified message identifier with accompanying data and MsgInfo fields. A ready signal from Training_Mgmt back to the LTSM indicates acceptance of each outbound message request.The second is the RDI message path, which carries link management messages between the RDI_SM within MainSM and the RDI_control block within SideBand_Top. Outbound RDI messages are presented by the RDI_SM as a message number and valid pulse in the lclk domain and cross into the clk_sb domain through a separate asynchronous path. Inbound RDI messages follow the symmetric path in the receive direction. This path is dedicated exclusively to RDI link management traffic and is routed separately from the LTSM training message path within SideBand_Top.The third group covers the SBINIT pattern control and the traffic handshake. During the SBINIT state, the LTSM drives pattern_mode, start_pat_req, and req_iter_count to SideBand_Top's Link_Controller to initiate the SBINIT sideband initialization pattern sequence; SideBand_Top returns iter_done and det_pat_rcvd to indicate completion and detection results. The traffic handshake — traffic_req from SideBand_Top to the RDI_SM and clk_handshake_done from the RDI_SM back to SideBand_Top — governs when the sideband upstream path may forward traffic through the RDI configuration channel. Additional control signals on this interface include phy_in_reset, which the LTSM asserts to SideBand_Top to cause the sideband register access engine to route all requests locally during reset, and pmo_en, which communicates the PMO negotiation outcome from MainSM to SideBand_Top's serializer. In the return direction, sticky_sb_pattern_detected carries the latched SBINIT pattern detection result, and sb_msg_timeout signals expiry of the 8 ms sideband message handshake timer.

3.4.2 Sideband ↔ Reg File Register Access Bus

The register access bus between SideBand_Top and Reg_File is a simple synchronous master-slave interface operating entirely in the clk_sb domain; no clock domain crossing is required. SideBand_Top's Reg_Access block acts as the sole master. It drives a 25-bit address (rf_addr), an 8-bit byte enable (rf_be), a 64-bit access width qualifier (rf_is_64b_access), a 64-bit write data word (rf_wdata), and separate read and write enable strobes (rd_en, wr_en) to Reg_File. Reg_File responds with a 64-bit read data word (rf_rdata) qualified by rdata_vld on the cycle the data is available, and with addr_err_o to signal an unsupported address, which the Reg_Access FSM uses to generate an Unsupported Request completion. Write transactions complete in the same cycle as the wr_en assertion; read transactions complete on the cycle rdata_vld is returned. No other subsystem drives or observes this bus; the interface is strictly point-to-point between Reg_Access and Reg_File.

3.4.3 MainSM ↔ Mainband

The interface between MainSM and unit_mb_die divides into a control plane driven from MainSM to unit_mb_die and a result plane driven in the reverse direction, both operating in the gated_lclk domain.

The control plane carries all signals through which the LTSM configures and enables the datapath for each training phase and for active operation. These include the LFSR operating state (mb_lfsr_state) that selects between clear, PRBS pattern, per-lane ID, and data modes; TX and RX width degradation codes (mb_width_deg_tx, mb_width_deg_rx); lane reversal enable (mb_reversal_en); mapper and demapper enables (mb_mapper_en, mb_demapper_en); receive deserializer enables (mb_rx_data_deser_en, mb_rx_valid_deser_en); clock pattern and embedded clock enables (mb_clk_pattern_en, mb_clk_embedded_en); clock pattern detector enable (mb_clk_detector_en); valid pattern enable (mb_valid_pattern_en); pattern comparator configuration including enable, mode, per-lane mask, iteration count, and clear (mb_pcmp_enable through mb_pcmp_clear); valid comparator configuration (mb_vcmp_enable, mb_vcmp_mode, mb_vcmp_clear); tri-state lane select codes for the clock, track, valid, and data signal groups (mb_tx_*_lane_sel); PLL speed selection (mb_pll_speed_sel); and the per-lane and aggregate error count thresholds used by the pattern comparator. The RDI_SM additionally drives rdi_lclk_g, which is the enable signal to the analog front-end mainband TX clock gate and is owned by the RDI gating logic rather than the LTSM.

The result plane carries the completion and pass/fail signals that the LTSM consumes to determine sub-state progression. These include pattern completion indicators for the LFSR training sequence (mb_lfsr_tx_done), the valid training pattern (mb_valid_done), and the clock pattern (mb_clk_done); pattern comparator done and per-lane pass vector (mb_pcmp_done, mb_pcmp_per_lane_pass); aggregate error flag (mb_pcmp_agg_error); valid comparator done and pass (mb_vcmp_done, mb_vcmp_pass); and the forwarded clock and track detection results (mb_clk_p_pass, mb_clk_n_pass, mb_track_pass). The valid frame error signal (mb_valid_frame_error) is also returned to the RDI_SM as pl_error, where it triggers an error response during active operation. unit_mb_die additionally forwards gated_lclk outward to both MainSM and SideBand_Top as the distributed parallel-domain clock reference.

3.4.4 Reg File ↔ MainSM

The interface between Reg_File and MainSM carries software-programmed configuration values from Reg_File to MainSM and hardware-driven status values from MainSM back to Reg_File. The two directions have different domain characteristics that reflect the asymmetric nature of register file operation: configuration values are written by software through the sideband access path and held as stable outputs in the clk_sb domain, while hardware status values are produced by combinational or registered outputs of the LTSM and RDI_SM in the lclk domain.

In the configuration direction, Reg_File drives the link training start control (phy_start_ucie_link_training_ctrl_out), target link width and link speed controls, x8 mode enable, clock phase and clock mode controls, PMO, TARR, L2SPD, and PSPT feature support controls, and

the per-lane comparison mask and D2C error threshold fields. These values are read by the LTSM and RDI_SM during their respective initialization and training sequences. Because they are quasi-static — written once during link setup and not changed during an active training sequence — the domain difference between clk_sb and lclk does not introduce functional hazards for these signals under normal operation.

In the status direction, MainSM drives the negotiated link width and speed enable status fields, the negotiated clock phase and clock mode status, the TARR, PMO, L2SPD, and PSPT feature enable status flags, the link status and link training or retraining indicator, and the busy flag. It also drives the error log capture inputs: the LTSM state at error capture (log0_state_n), the lane reversal and width degradation flags at the time of error, and the two preceding LTSM states for diagnostic context. The pl_trainerror assertion serves as the capture enable that causes Reg_File to latch these values into the error log registers. Since these status updates occur at well-defined training event boundaries rather than continuously, the capture is treated as a controlled hardware write rather than a continuously synchronized value.

3.5 Top-Level Parameters and Capability Straps

The UCIe_PHY module exposes two categories of compile-time constants at its parameter boundary. The first category, configurable parameters, controls the structural and timing dimensions of the design: data widths, lane counts, packet granularities, and timing constants used in simulation and in hardware timer calculations. The second category, static capability straps, establishes the PHY's capability profile as it is advertised to the remote link partner through the register file and the MBINIT.PARAM negotiation sequence. Both categories are declared as SystemVerilog parameters at the UCIe_PHY level and propagated through the subsystem hierarchy. Neither can be changed at runtime; they are fixed at integration time and determine the behavioral envelope within which the four subsystems operate.

3.5.1 Configurable Parameters

DATA_WIDTH_MB (default 32) defines the width of each parallel lane word in the mainband serialization and deserialization path. At the DDR serialization ratio of 16 described in Section 3.3.1, a 32-bit parallel word corresponds to one complete set of serialized bits per lane per parallel clock cycle. DATA_WIDTH_SB (default 64) defines the width of the sideband packet word in the parallel clk_sb domain, consistent with the UCIe specification's requirement that every sideband packet, whether a header or a data payload, occupies exactly 64 serial bit times.

NUM_LANES (default 16) sets the number of mainband data lanes instantiated in unit_mb_die. For this Standard Package implementation, 16 is the full lane count; x8 operating width is achieved at runtime by the width degradation logic rather than by reducing the instantiated lane count. N_BYTES (default 64) defines the byte width of the RDI data path presented to the Adapter, such that the total parallel data bus width is 8 × N_BYTES = 512 bits, matching the product of 16 lanes and 32 bits per lane.

GAP_WIDTH (default 32) defines the number of sideband serializer idle cycles inserted between consecutive outbound sideband packets in standard (non-PMO) mode, implementing the inter-packet gap required by the sideband protocol. VALID_PATTERN (default 32'h0F0F0F0F) encodes the expected 32-bit framing pattern — four unit intervals asserted followed by four de-asserted, repeated — used by the valid frame detector on the receive side to locate deterministic 8-UI frame boundaries.

PLL_PERIOD_NS (default 0.5 ns) and CLK_FRQ_HZ (default 62.5 MHz) are timing constants used within the design's internal timer calculations. CLK_FRQ_HZ is consumed by the RDI_SM's unit_gating_logic and the LTSM's timeout infrastructure to derive cycle counts for time-bounded operations, including the 8 ms sideband message handshake timeout and the 1 µs ungating delay. PLL_PERIOD_NS is used by the clock pattern generator for timing reference during MBINIT.REPAIRCLK. RX_ALIGN_DELAY (default 2) accounts for the pipeline depth of the deserializer output stage and is used to offset the capture alignment in the receive data path.

3.5.2 Static Capability Straps

The capability straps are single-bit or multi-bit parameters that populate the hardware-initialized (HWInit) fields of the UCIe Link Capability and PHY Capability registers and are simultaneously supplied as inputs to the LTSM during the MBINIT.PARAM capability advertisement exchange. Because these fields are HWInit, they are loaded at domain reset from the strap values and are not modifiable by software; a remote die reading the corresponding register fields therefore observes the values fixed at integration time.

MAX_LINK_WIDTH_CAP (default 3'd0) encodes the maximum supported link width in the UCIe-defined three-bit encoding, where 0 corresponds to x16. MAX_LINK_SPEED_CAP (default 4'h5) encodes the maximum supported IO link speed, where the value 5 corresponds to 32 GT/s in the UCIe Link Capability speed encoding. SPMW_CAP (default 0) is the Standard Package Module Width strap: when cleared it declares the module as a full x16 module; when set it declares it as an x8 module or a x16 module constrained to operate at x8 width. SPMW_CAP is supplied both to the Reg_File to populate the SPMW HWInit field and directly to the LTSM, where it constrains the lane width negotiation outcome during MBINIT.

PMO_CAP (default 1) declares whether Sideband Performant Mode Operation is supported by this implementation. When set, it is advertised in the UCIe Link Capability register and presented as the local capability input to the MBINIT.PARAM state, which negotiates PMO enablement with the remote die. The remaining feature capability straps — PSPT_CAP, L2SPD_CAP, and TARR_CAP — are all defaulted to zero, consistent with the exclusion of Priority Sideband Packet Transfer, L2 Sideband Power Down, and Tx Adjustment during Runtime Recalibration from this implementation's scope.

SUPPORTEDVSWING_CAP (default 5'h01), CLK_MODE_CAP (default 2'b10), and CLK_PHASE_CAP (default 2'b00) encode the PHY's transmit voltage swing support, receive clock mode preference, and receive clock phase capability respectively. These fields are advertised during MBINIT.PARAM and populate the corresponding PHY Capability register fields. CLK_MODE_CAP defaulting to 2'b10

declares continuous clock mode support; CLK_PHASE_CAP defaulting to 2'b00 declares differential clock operation. ADVANCED_PKG_CAP (default 0) identifies this instantiation as a Standard Package PHY, driving the package type field in the PHY Capability register. MODULE_ID (default 2'b00) carries the module identifier included in the MBINIT.PARAM message; it defaults to zero and is relevant only in multi-module configurations, which are outside the scope of this implementation.

4.0 Mainband Design

4.1 Subsystem Overview

4.1.1 MainBand Top-Level Structure

The mainband subsystem is implemented as unit_mb_die, a structurally complete die-level module that integrates the full transmit and receive datapaths of the UCIe physical layer into a single top-level boundary. The module is organized around a digital half (digital_mb) and an analog hard macro (mainband_analog_hard_macro), with a precisely defined parallel-domain interface between them. All data processing — byte-to-lane mapping, LFSR scrambling and descrambling, lane reversal, width degradation, valid framing generation and detection, pattern generation and comparison, and the clock pattern generation and detection functions — resides in the digital half. The analog hard macro owns the PLL, the clock divider chain, the per-lane serializers and deserializers, and the tri-state lane output buffers.

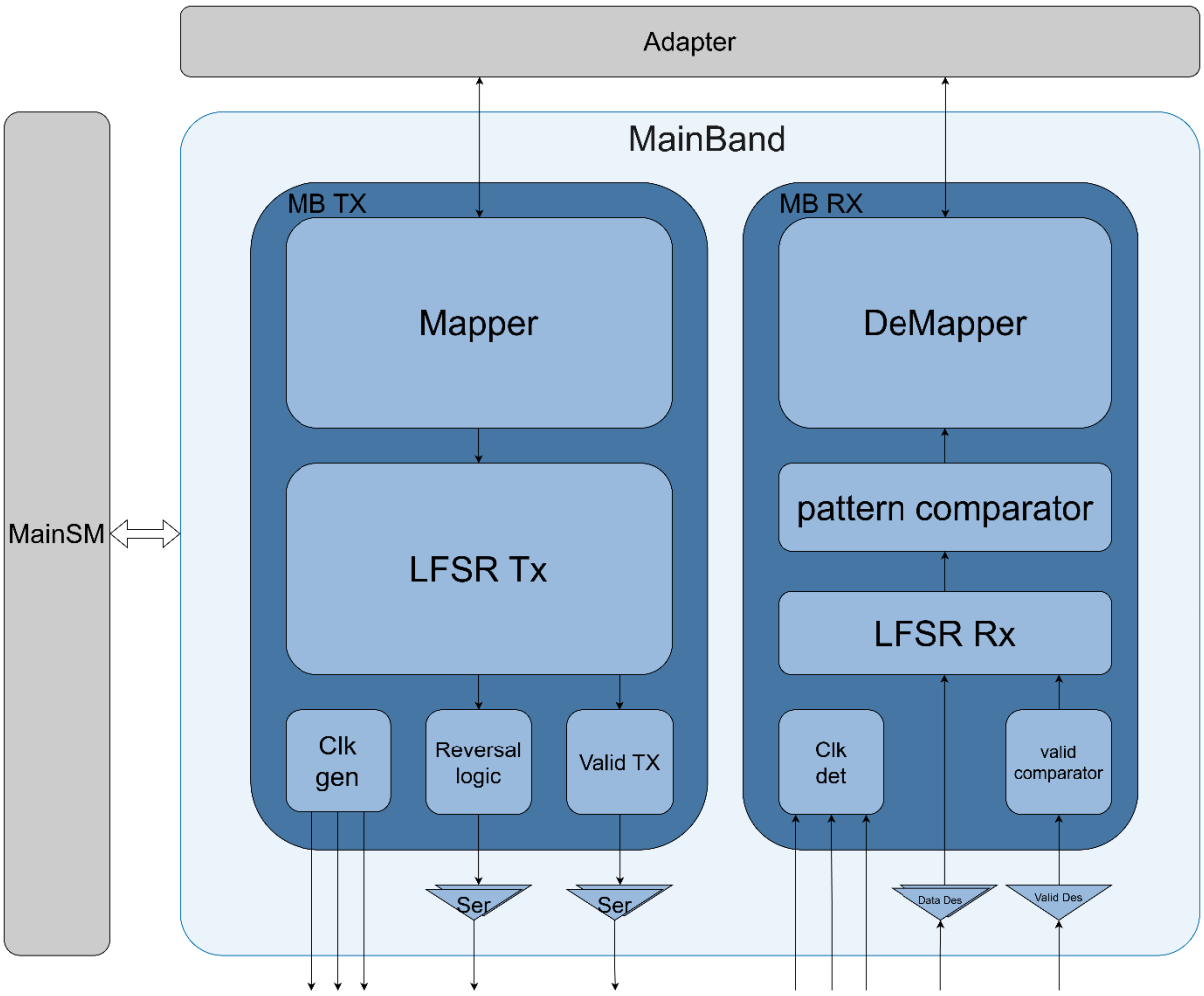


Figure 4-1: MainBand Top-Level Structure

The transmit path carries protocol data downward from the D2D Adapter through three functional planes. The data plane accepts the 512-bit adapter word on lp_data, distributes it across the active lane set through the Mapper, applies the LFSR-selected processing mode (scrambling, PRBS-23 pattern, or per-lane ID sequence) through LFSR_TX, optionally reverses the physical lane order through the lane reversal array, and presents 32-bit parallel words to the analog serializers. The valid plane generates the UCIe valid framing signal through unit_valid_tx and presents its parallel encoding to the valid serializer. The clock plane drives the forwarded differential clock and track lane signals through unit_clk_pattern_gen_tx, which supports both 128-UI clock burst mode for training and continuous embedded-clock mode for active data transfer. The receive path mirrors this organization: the analog deserializers and valid deserializer recover parallel words from the incoming serial streams, LFSR_RX descrambles and width-degrades the data lanes, the pattern comparator and valid comparator execute the training qualification measurements, the clock pattern detector validates the received forwarded clock, and the Demapper reassembles the recovered lane words into a 512-bit protocol word delivered to the D2D Adapter.

The subsystem is a purely datapath-oriented block with no autonomous control. All mode-select, enable, and training-phase signals are received from MainSM over the control plane of the MainSM-to-Mainband interface. All training results — LFSR sequence completion, valid pattern completion, clock pattern completion, per-lane pattern comparator pass vectors, aggregate error flags, valid comparator outcomes, and forwarded clock and track detection results — are returned from unit_mb_die to MainSM over the result plane of the same interface. This strict separation ensures that the training sequencing logic defined by the LTSM state machine remains entirely within MainSM and is not distributed across the datapath. On its upper boundary, unit_mb_die exchanges the mainband parallel data words directly with the D2D Adapter via the RDI data path. On its lower boundary, it presents pre-serialization parallel lane words to the analog hard macro and receives recovered parallel lane words from it.

The clocking structure within unit_mb_die is built around the analog PLL. The PLL output (high-speed serialization clock) feeds the serializer serial ports directly and is divided by sixteen through a clock divider to produce lclk, the raw parallel-domain clock. A glitch-free clock gate produces gated_lclk from lclk under the enable signal controlled by the RDI_SM gating logic. gated_lclk is the sole clock of all digital parallel-domain logic within unit_mb_die, including the Mapper, LFSR_TX, unit_valid_tx, and the parallel-load interfaces of the serializers. It is also forwarded outward to MainSM and SideBand_Top as the distributed mainband parallel-domain clock reference. The PLL speed selection (i_pll_speed_sel) is driven from MainSM at configuration time and selects among the supported link rates. The receive path samples incoming data on the forwarded clock received from the remote die, reconstructed inside the analog hard macro into the parallel-domain i_data_valid qualifier.

4.2 Transmit (TX)

The MB_TX is the transmit subsystem of the UCIe physical layer. It receives 512-bit parallel data from the Adapter layer and drives it out as high-speed serial streams across 16 physical lanes, together with a forwarded clock and a valid framing signal. The subsystem is organized into three planes:

Data Plane — The Mapper distributes the 512-bit adapter word evenly across 16 lanes of 32 bits each, supporting width-degraded modes when lanes are faulty. LFSR_TX then processes the lanes in two modes: during normal operation it scrambles each lane by XOR-ing with an LFSR sequence

to eliminate DC wander; during link training it repurposes the same hardware to generate PRBS-23 test patterns and Per-Lane ID sequences. The optional mb_tx_reversal block corrects any physical lane-order swap introduced by the package substrate. Data_ser finally serialises each 32-bit lane word into a single high-speed serial bit stream, producing 16 serial outputs at the PLL bit rate.

Valid Plane — Valid_TX generates the UCIe VALID framing signal — four UI asserted followed by four UI de-asserted — on a dedicated physical lane, giving the receiver a deterministic 8-UI boundary to locate data frames. Valid_ser serialises this pattern onto the TVLD_P pin in the same way Data_ser handles the data lanes.

Clock Plane — A single PLL synthesises pll_clk, the high-speed bit-rate clock used directly by the serial sides of the serializers and by clk_pattern_gen_TX. For the parallel digital logic, pll_clk is divided by 16 in clk_div/16 to produce div_clk, which is then passed through clk_gate. clk_gate applies a glitch-free LTSM-controlled enable to output mb_clk, the slower clock that drives Mapper, LFSR_TX, Valid_TX, and the parallel-load sides of both serializers. When the link is idle the LTSM stops mb_clk, eliminating dynamic power in all parallel-domain logic. clk_pattern_gen_TX drives the forwarded differential clock pins (TCKP_P / TCKN_P) and track lane (TTRK_P) for source-synchronous capture at the receiver, and transmits a 128-iteration continuity test pattern during the MBINIT.REPAIRCLK training state.

Link Training — The LTSM orchestrates all blocks through dedicated control signals, reconfiguring the same physical hardware across every training phase — clock repair, PRBS exchange, lane identification, and speed ramp-up — before handing off to the normal data path for ACTIVE operation.

4.2.1 Mapper:

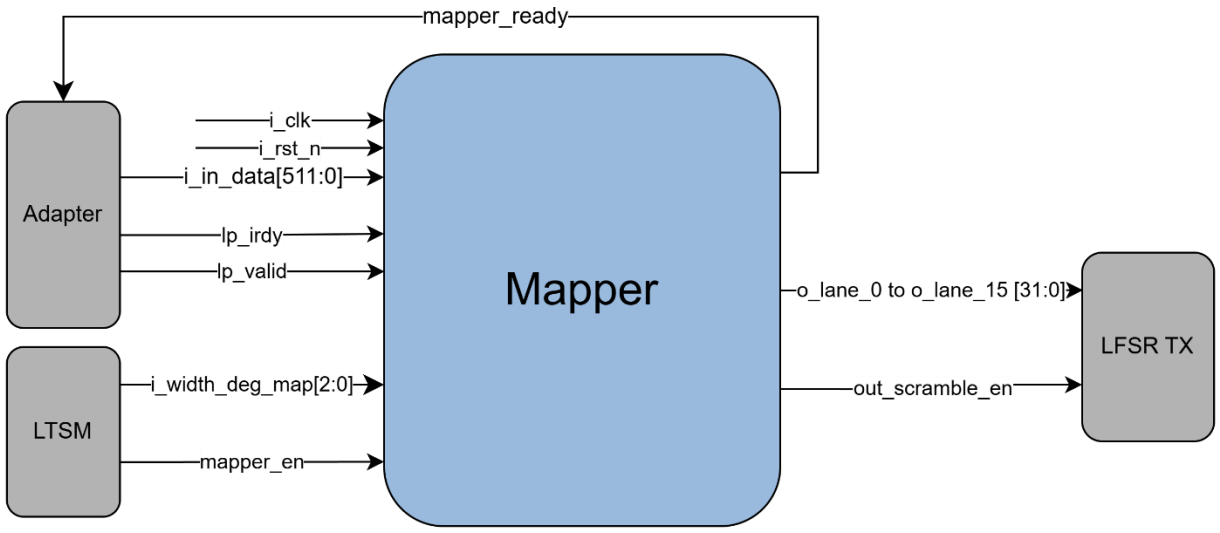


Figure 4-2: Mapper Block Diagram

The Mapper operates on the TX side and is the entry point of the mainband transmit path. Its primary role is to receive 512-bit (64-byte) wide parallel data from the Adapter layer at each clock cycle and distribute it evenly across 16 physical lanes, each 32 bits wide. Receives 512-bit (64-byte) data from the Adapter RDI interface.Splits the data into 16 independent 32-bit lane streams. Supports dynamic lane-width configurations (x16, x8, x4 degradation modes) controlled by the LTSM. When disabled (by LTSM outside active data phases), all outputs are forced to zero. Lane selection / width degradation is controlled by the i_width_deg_mapper signal.

UI \ Lane	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0 - 7	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15
8 - 15	B16	B17	B18	B19	B20	B21	B22	B23	B24	B25	B26	B27	B28	B29	B30	B31
16 - 23	B32	B33	B34	B35	B36	B37	B38	B39	B40	B41	B42	B43	B44	B45	B46	B47
24 - 31	B48	B49	B50	B51	B52	B53	B54	B55	B56	B57	B58	B59	B60	B61	B62	B63
32 - 39	B64	B65	B66	B67	B68	B69	B70	B71	B72	B73	B74	B75	B76	B77	B78	B79
40 - 47	B80	B81	B82	B83	B84	B85	B86	B87	B88	B89	B90	B91	B92	B93	B94	B95
48 - 55	B96	B97	B98	B99	B100	B101	B102	B103	B104	B105	B106	B107	B108	B109	B110	B111
56 - 63	B112	B113	B114	B115	B116	B117	B118	B119	B120	B121	B122	B123	B124	B125	B126	B127
64 - 71	B128	B129	B130	B131	B132	B133	B134	B135	B136	B137	B138	B139	B140	B141	B142	B143
72 - 79	B144	B145	B146	B147	B148	B149	B150	B151	B152	B153	B154	B155	B156	B157	B158	B159
80 - 87	B160	B161	B162	B163	B164	B165	B166	B167	B168	B169	B170	B171	B172	B173	B174	B175
88 - 95	B176	B177	B178	B179	B180	B181	B182	B183	B184	B185	B186	B187	B188	B189	B190	B191
96 - 103	B192	B193	B194	B195	B196	B197	B198	B199	B200	B201	B202	B203	B204	B205	B206	B207
104 - 111	B208	B209	B210	B211	B212	B213	B214	B215	B216	B217	B218	B219	B220	B221	B222	B223
112 - 119	B224	B225	B226	B227	B228	B229	B230	B231	B232	B233	B234	B235	B236	B237	B238	B239
120 - 127	B240	B241	B242	B243	B244	B245	B246	B247	B248	B249	B250	B251	B252	B253	B254	B255

Figure 4-4: Byte to Lane mapping for Standard package x16

Lane	8	9	10	11	12	13	14	15
	or							
UI \ Lane	0	1	2	3	4	5	6	7
0 - 7	B0	B1	B2	B3	B4	B5	B6	B7
8 - 15	B8	B9	B10	B11	B12	B13	B14	B15
16 - 23	B16	B17	B18	B19	B20	B21	B22	B23
24 - 31	B24	B25	B26	B27	B28	B29	B30	B31
32 - 39	B32	B33	B34	B35	B36	B37	B38	B39
40 - 47	B40	B41	B42	B43	B44	B45	B46	B47
48 - 55	B48	B49	B50	B51	B52	B53	B54	B55
56 - 63	B56	B57	B58	B59	B60	B61	B62	B63

Figure 4-3: Byte to Lane mapping for Standard package x16 degraded to x8

Table 4-1: Mapper Interface Signals

Signal Name	Direction	Width	Description
i_clk	Input	1	System clock (mb_clk)
i_rst_n	Input	1	Active-low synchronous reset
lp_valid	Input	1	Adapter to Physical Layer indication that data is valid on the corresponding (lp_data) bytes
lp_irdy	input	1	Adapter to Physical Layer signal indication that the Adapter has data to send. It is asserted when lp_valid is asserted.
lp_data[511:0]	Input	512	64-byte payload from Adapter to Physical Layer.
i_width_deg_mapper	Input	3	Lane-width degradation select per UCIe (x16, x8, x4 degradation modes) (set by LTSM at MBTRAIN.REPAIR)
mapper_en	Input	1	Enable signal from LTSM; when low all outputs are forced to zero
o_lane_0 … o_lane_15	Output	32 each	16 parallel lane data outputs to LFSR_TX (scrambler stage)
scramble_en	Output	1	Enable signal from The Mapper to the LFSR_TX for the scrambler stage
pl_trdy	Output	1	Physical Layer to the Adapter Layer indicates That the Physical Layer is ready to accept data.

4.2.2 LFSR TX

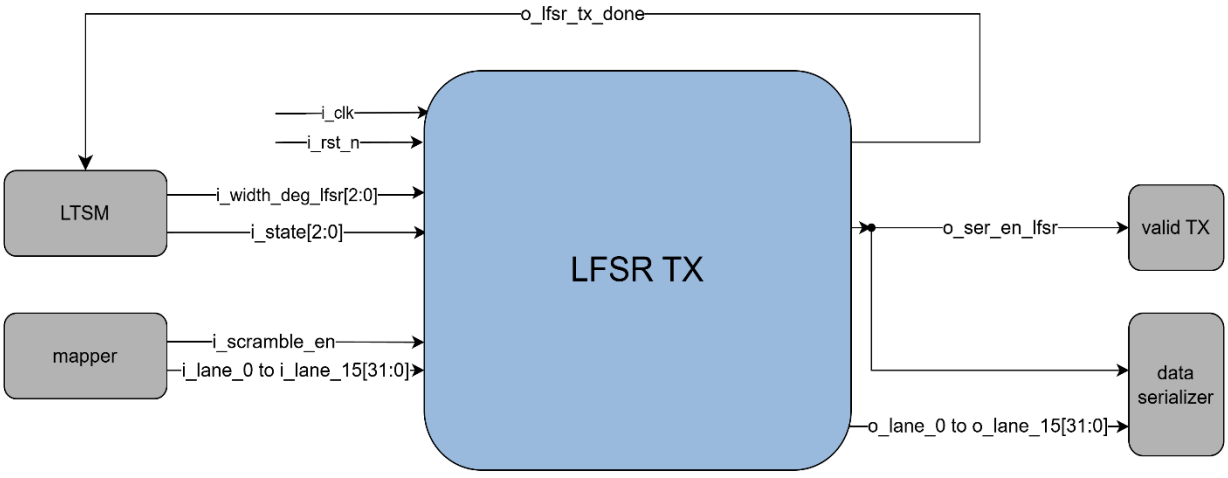


Figure 4-5: LFSR Tx diagram

LFSR_TX is the most complex and versatile block in the MB_TX datapath. It is responsible for two fundamentally different but hardware-unified functions: generating PRBS test patterns during link training and scrambling real payload data during normal (ACTIVE) operation. The design reuses the same LFSR state machine for both tasks.

The block also hosts the per-lane ID generator, which produces the specific 128-iteration training patterns required by UCIe for lane identification, deskew, and valid-lane confirmation during the MBTRAIN.LANEREPAIR and MBTRAIN.SPEED states.

LFSR Architecture

The block contains one LFSR instance per active lane (up to 16), each implementing the PRBS-23 polynomial G(X) = $x^{23} + x^{21} + x^{16} + x^{8} + x^{5} + x^{2} + 1$. Each LFSR produces a 32-bit pseudo-random word per clock cycle by unrolling 32 steps of the recurrence relation in combinational logic. This provides both a compact hardware footprint and the ability to generate data at full parallel bus width without multi-cycle latency.

Training Mode — PRBS-23 Pattern Generation

During link training states (MBTRAIN),

The LTSM asserts i_state = PATTERN_LFSR and sets data_lane_mode to select the LFSR output path. The LFSR runs freely and generates a continuous PRBS-23 sequence on all enabled lanes. The receiver uses this known sequence to measure bit-error rate, find lane alignment, and optimise equalisation settings.

At the start of each new training state the LFSRs are seeded with a deterministic initial value so that the transmitter and receiver sequences are phase-aligned, enabling the receiver to detect the first valid symbol without requiring explicit seed exchange.

Active (Data) Mode — Scrambling

During normal data transmission (LTSM ACTIVE state), The LTSM asserts i_state = DATA_TRASFER, scramble_en is asserted from the Mapper, The LFSR runs in parallel with the Mapper output. The scrambler performs a bitwise XOR between the incoming 32-bit lane data and the current 32-bit LFSR state: data_out = data_in XOR LFSR_value. This operation randomises the transmitted bit stream.

Per-Lane ID Generator

The LTSM asserts i_state = PER_LANE_IDE which generates the 128-iteration Per-Lane ID (PLID) patterns required by UCIe. Each lane transmits a unique pattern that allows the receiver to identify which physical lane it is observing, supporting lane-to-lane mapping even when FAB routing swaps lane order. The block integrates with Valid_TX to assert the correct valid framing around each PLID burst. MUX_1 selects between LFSR/scrambler output and per_lane_id_gen output based on data_lane_mode.

The LFSR_TX asserts ser_en_lfsr to the Data serializer (to serialize the Data) and The Valid TX (to frame the Data).

Table 4-2: LFSR Tx Interface Signals

Signal Name	Direction	Width	Description
i_clk	Input	1	System clock (mb_clk) driving all LFSR_TX logic
i_rst_n	Input	1	Active-low reset
i_width_deg_lfsr	Input	3	Lane-width degradation select per UCIe (x16, x8, x4 degradation modes) (set by LTSM at MBTRAIN.REPAIR)
scramble_en	Input	1	Mapper to LFSR_TX control signal which Enables the XOR scrambler in ACTIVE mode.
i_state	Input	3	LTSM control signal which indicates the mode of operation (IDLE, CLEAR_LFSR, PATTERN_LFSR, PER_LANE_IDE, DATA_TRANSFER)
i_lane_0 … i_lane_15	Input	32 each	Parallel mapper output lanes (32 bits per lane)
o_lane_0 … o_lane_15	Output	32 each	Processed (scrambled or PRBS or Per lane ID) lane data to Data serializer
ser_en_lfsr	Output	1	Valid Frame and Data serializer enable derived from LFSR_TX
o_lfsr_tx_done	Output	1	Indicates completion of LFSR training pattern sequence, Per lane ID pattern to LTSM

4.2.3 Valid Lane Transmitter

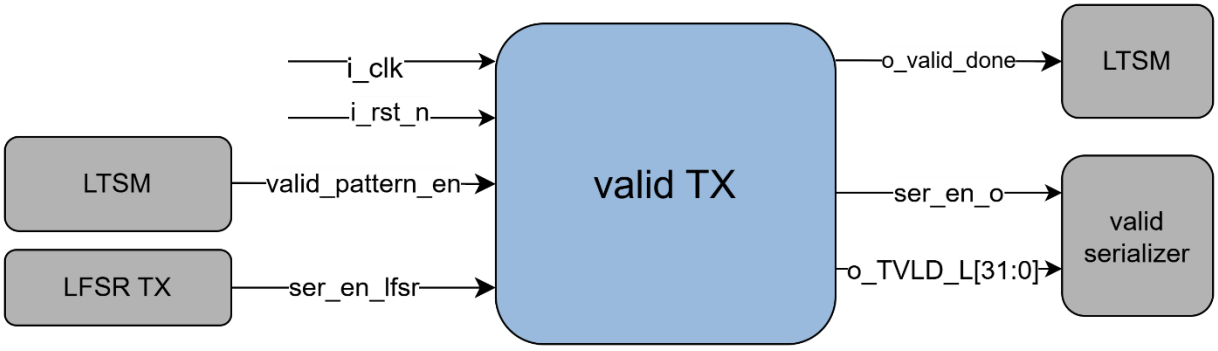


Figure 4-6: Valid Tx daigram

Valid_TX (also referred to as the Valid Controller or Valid Generator) generates and controls the VALID lane signaling towards the serializer in both training and active transmission modes.

- Produces a fixed 32-bit repeating pattern: four consecutive 1s followed by four consecutive 0s (0x0F0F0F0F), compliant with UCIe framing requirements.
- During training, outputs 128 iterations of the VALTRAIN pattern (TVLD_L) with the forwarded clock to support link synchronization.
- During active operation, the 4-UI / 4-UI pattern provides deterministic valid framing aligned to 8-UI boundaries.
- Controlled by both the LTSM (valid_pattern_en) and the LFSR_TX block (ser_en_lfsr).
- Reports pattern completion to the LTSM via o_done.

Table 4-3: Valid Tx Interface signals

Signal Name	Direction	Width	Description
i_clk	Input	1	System clock (mb_clk) synchronizing all Valid_TX operations
i_rst_n	Input	1	Active-low reset; initializes the block to IDLE state
valid_pattern_en	Input	1	LTSM enable: activates training-mode VALTRAIN pattern output
ser_en_lfsr	Input	1	LFSR_TX enable: triggers active framing mode (ser_en_lfsr)
o_TVLD_L[31:0]	Output	32	32-bit parallel VALID pattern to serializer; 128 iterations of four-1s / four-0s
o_valid_done	Output	1	Signals to LTSM that the VALTRAIN pattern transmission is complete
ser_en	Output	1	Enable signal to the Valid serializer

4.2.1 Lane Reversal

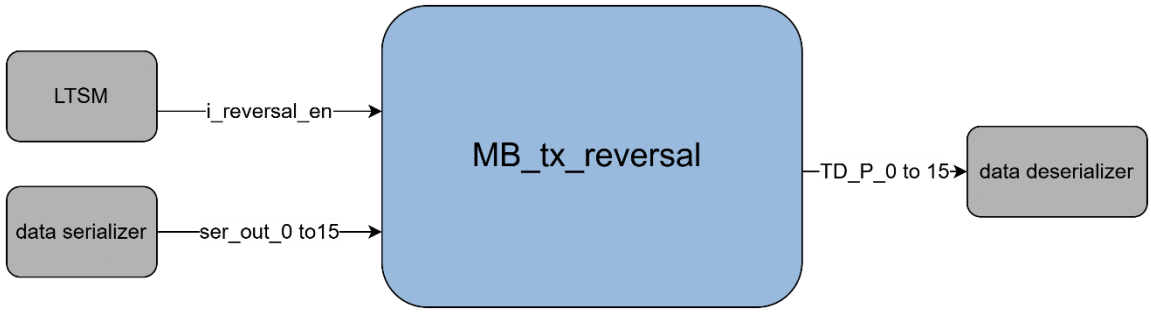


Figure 4-7: MB Reversal diagram

The mb_tx_reversal block handles optional lane/bit reversal of the 16-lane parallel data bus before serialization. In a UCIe standard package, the physical lane order during Fabrication may be reversed with respect to the logical lane numbering. The reversal block corrects this transparently in hardware.

- Receives 16 x single bit lanes from Data serializer.
- When reversal_en is asserted, re-maps lane N to lane (15-N) and optionally reverses the bit order within each lane.
- When reversal_en is de-asserted, the block operates as a straight-through pass.
- Supports both x16, x8 and x4 lane-reversal modes as required by UCIe MBTRAIN.

Table 4-4: Lane Reversal Interface Signals

Signal Name	Direction	Width	Description
reversal_en	Input	1	LTSM control signal which Enables lane/bit reversal; de-asserted = straight through
ser_out_0 … ser_out_15	Input	1 each	Parallel lanes data serialized from Data serializer
TD_P_0 … TD_P_15	Output	1 each	Optionally reversed lane data

4.2.2 Clock Pattern Generator

The clk_pattern_gen_TX block generates the UCIe forwarded differential clock pattern on the TCKP_L / TCKN_L clock lanes, and the track-lane pattern on TTRK_L. It is used during the MBINIT.REPAIRCLK training state and also provides the forwarded clock in normal operation.

- Transmits an unscrambled clock repair pattern for 128 iterations of(16 clock cycles followed by 8 cycles of low) on TCKP_L, TCKN_L, and TTRK_L during MBINIT.REPAIRCLK.

- In normal link operation, continuously forwards the PLL-derived clock to the remote receiver.

  clk_embedded_en enables the free running clks which the remote partner uses this clock to sample data at its receiver.

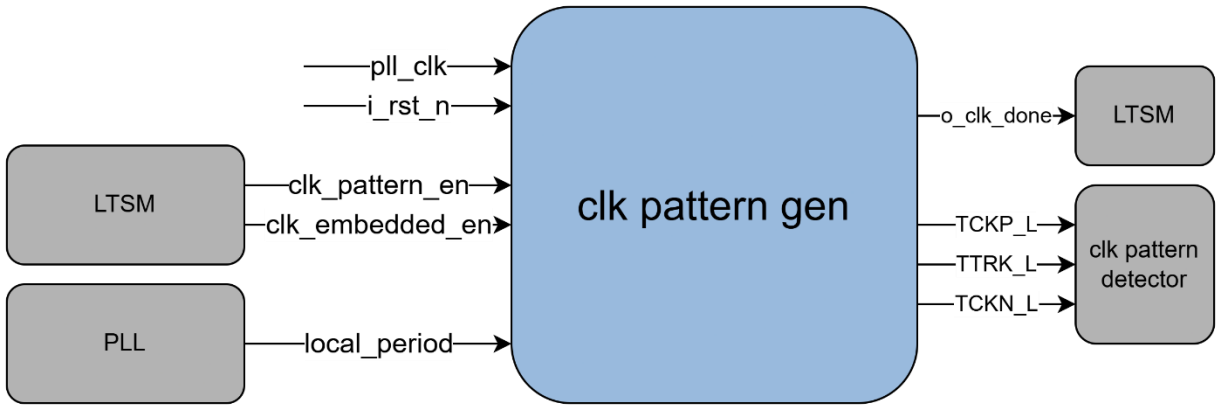


Figure 4-8: CLK Pattern Generator

Table 4-5: Clock Pattern Generator Interface Signals

Signal Name	Direction	Width	Description
i_rst_n	Input	1	Active-low synchronous reset (synchronous to pll_clk)
pll_clk	input	1	High-speed PLL bit clock
clk_embedded_en	Input	1	LTSM enables embedded-clock mode.
clk_pattern_en	Input	1	LTSM enable: start clock-repair pattern transmission
Local_period	input	N	Indicates the period of the pll clk
TCKP_L	Output	1	Positive leg of differential forwarded clock (UCIe module output)
TCKN_L	Output	1	Negative leg of differential forwarded clock
TTRK_L	Output	1	Track lane output for per-lane deskew at the receiver
o_clk_done	Output	1	Signals 128-iteration completion to LTSM

4.2.3 Data Lane Serializer

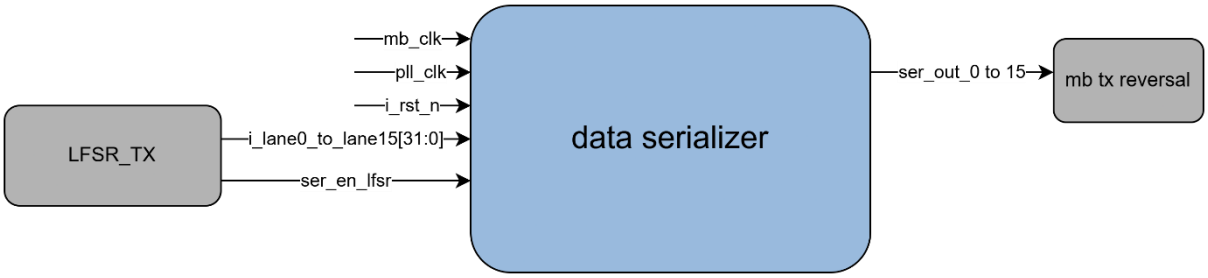


Figure 4-9: Data Lane Serializer Diagram

Data serializer converts 16 independent 32-bit parallel lane words into 16 high-speed single-bit serial streams, one per physical data lane. It is a parallel-to-serial (P2S) converter clocked at the PLL output frequency.

- Accepts 16 × 32-bit inputs from the LFSR_TX block.
- Each 32-bit word is shifted out LSB-first in 32 consecutive UI periods on its corresponding serial output pin.
- Clocked by pll_clk (high-speed PLL clock) and gated by the gated_clk (mb_clk) from clk_gate.
- Enabled by ser_en_lfsr from LFSR_TX block.

Table 4-6: Data Lane Serializer Interface Signals

Signal Name	Direction	Width	Description
mb_clk	Input	1	System clock (mb_clk)
pll_clk	Input	1	High-speed PLL clock used as serialization bit clock
i_rst_n	Input	1	Active-low synchronous reset
i_lane_0 … i_lane_15	Input	32 each	32-bit parallel lane data from LFSR_TX
ser_en_lfsr	input	1	Enable signal for the Data serializer from LFSR_TX block
ser_out_0 … ser_out_15	Output	1 each	High-speed serial data outputs (one per physical lane)

4.2.4 Valid Lane Serializer

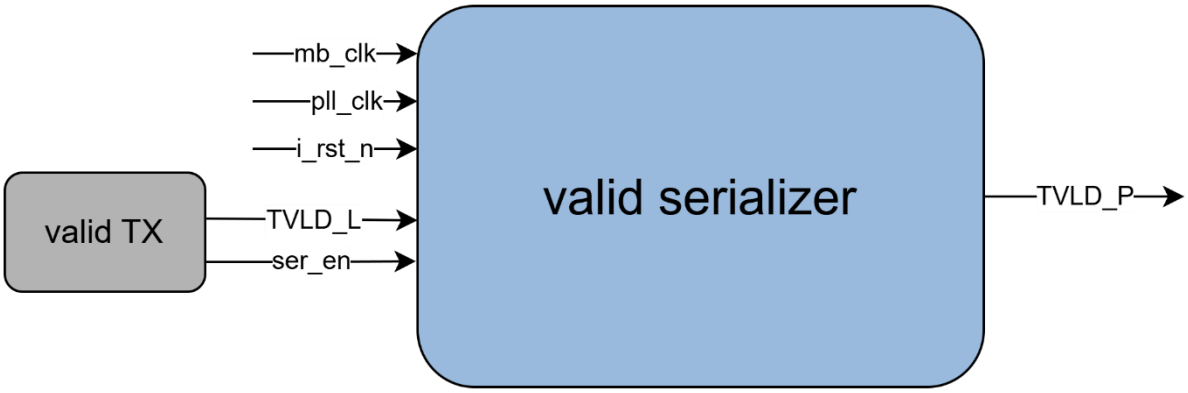


Figure 4-10: Valid Lane Serializer Diagram

Valid serializer converts the 32-bit parallel VALID pattern from Valid_TX into a high-speed serial TVLD_L stream. It mirrors the architecture of Data_ser but operates on the single VALID lane rather than the 16 data lanes.

- Accepts 32-bit parallel valid pattern (o_TVLD_L) from Valid_TX.
- Serializes the pattern onto a single serial pin (ser_out / TVLD_L).
- Clocked by pll_clk and gated_clk.
- o_TVLD_L output is physically the UCIe TVLD_L pin on the module.

Table 4-7: Valid Lane Serializer Interface Signal

Signal Name	Direction	Width	Description
mb_clk	Input	1	Gated PLL clock (mb_clk) from clk_gate
pll_clk	Input	1	High-speed PLL clock as bit clock
i_rst_n	Input	1	Active-low synchronous reset
ser_en	input	1	Enable signal for the Valid serializer from Valid_TX block
TVLD_L	Input	32	32-bit parallel VALID pattern from Valid_TX
TVLD_P	Output	1	Serialized VALID signal — maps to the UCIe TVLD_L physical pin

4.2.5 Clk Gate

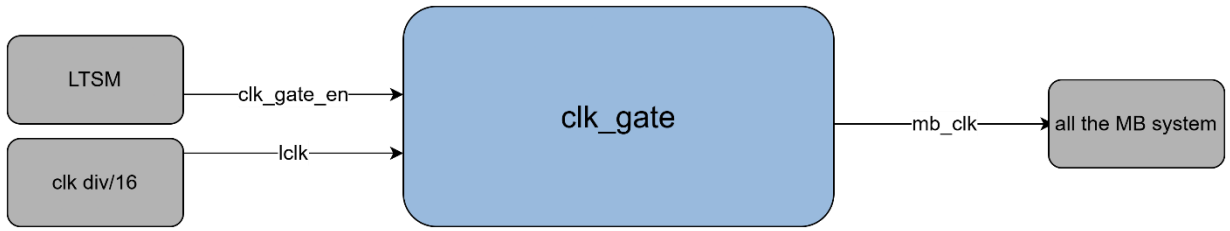


Figure 4-11: CLK Gate Diagram

The clk_gate block implements an integrated clock gating (ICG) cell for the mainband slow-clock domain. It sits between the clk_div/16 output (div_clk) and all the mainband digital logic blocks. Its output, mb_clk, is the clock that drives Mapper, LFSR_TX, Valid_TX, and the parallel-load sides of Data_ser and Valid_ser. It does NOT gate the high-speed pll_clk used by the serializer shift registers — pll_clk always runs freely from the PLL and is distributed to the serial sides of Data_ser and Valid_ser directly.

- Takes div_clk (from clk_div/16) and a clock-gate enable signal (clk_gate_en from LTSM) as inputs.
- Outputs a glitch-free gated clock (mb_clk) that drives all mainband parallel-domain blocks: Mapper, LFSR_TX, Valid_TX, and the parallel-load sides of Data_ser and Valid_ser.

Table 4-8: Clk Gate Interface Signals

Signal Name	Direction	Width	Description
clk_gate_en	Input	1	Clock gate enable from LTSM. mb_clk is held at 0 and all mainband parallel-domain logic is frozen. When high, div_clk passes through as mb_clk.
div_clk	Input	1	Slow divided clock from clk_div/16 (= pll_clk divided by 16). This is the clock source to be gated. It runs continuously as long as the PLL is locked and clk_div/16 is not in reset.
mb_clk	Output	1	Glitch-free gated mainband clock. Identical to div_clk when clk_gate_en is asserted; held at 0 otherwise. Distributed to Mapper, LFSR_TX, Valid_TX, Data_ser and Valid_ser.

4.3 Receive (RX)

The MB_RX_TOP module is the top-level receive subsystem of the UCIe physical layer, responsible for capturing high-speed serial inputs and converting them back into parallel protocol data. It operates in two principal modes: link training, during which pattern comparison and lane qualification are performed, and active data transfer, during which descrambled payload data is reconstructed and forwarded to the upper protocol layer. The module is structured into the following distinct functional planes:

Data Plane: High-speed serial data lanes (i_RD_P_0_15_lane) are captured by the Data Deserializer array (DATA_DES), which produces parallel lane words (O_PAR_DATA_0_15_lane). The LFSR_RX block then descrambles these words, handles lane-width degradation (i_width_deg_rx) across x16, x8, and x4 link widths, and routes data to the appropriate downstream block depending on the active mode. During active data transfer, descrambled streams are forwarded to the DEMAPPER to reconstruct the final 512-bit output (o_out_data) and assert o_pl_valid. During link training, the locally generated reference patterns (O_LFSR_FINAL_GEN) and received lane data (O_RX_LANE) are forwarded to the PATTERN_COMPARATOR for error tracking against thresholds (I_PCMP_THR_PER_LANE, I_PCMP_THR_AGG), producing comparison status (O_PCMP_DONE, O_PCMP_PER_LAN_PASS, O_PCMP_AGG_ERROR).

Valid Plane: The serialized valid signal (i_RVLD_P) is sampled by the Valid Deserializer (VALID_DES), which outputs the deserialized framing word (VALID_FRAME_DATA / VALID_FRAME_VALID) and frame-alignment signals to the VALID_RX (Frame Detector). The Frame Detector compares the incoming parallel stream against the expected 8-UI framing pattern and, upon a match, asserts O_VALID_FRAME_PULSE to simultaneously trigger frame-aligned capture across all 16 DATA_DES lanes, ensuring the entire data plane is locked to the same boundary. The deserialized framing data is also routed to the VALID_COMPARATOR, which evaluates framing signal health against configuration thresholds (i_VCMP_thr) and reports pass/fail status via O_VCMP_DONE and error flags.

Clock & Diagnostics Plane: The incoming differential clock (I_RCKP_P), its complement (I_RCKN_P), and the track lane (I_RTRK_P) are routed to the CLK_PATTERN_DETECTOR to validate early clock behavior using the local I_PLL_CLK, outputting status flags (O_CLK_P_PASS, O_CLK_N_PASS, O_TRACK_PASS). For normal deserialization, the parallel logic is clocked by I_MB_CLK while high-speed sampling relies on the phase-aligned clock domain.

4.3.1 Valid Lane Deserializer

The unit_valid_deserializer is a DDR deserializer for the UCIe Main-Band RX physical layer. It captures the high-speed serial valid framing signal (ser_data_in / i_RVLD_P) on the RCKP_P clock domain, deserializes it to a 32-bit parallel word, and synchronously transfers the result to the i_mb_clk domain. The free-running shift register state is exposed on o_shift_reg for frame pattern detection, and o_count_16 indicates when a complete 32-bit frame boundary is reached. Deserialized output is driven on o_valid_frame_data with o_valid_frame_vld qualifying each new word. Activation is controlled by i_rx_valid_dese_en; reset by i_rst_n.

Sampling is performed on both edges of RCKP_P (DDR mode), accumulating two bits per clock cycle until a full 32-bit word is formed. The i_valid_pulse input qualifies write operations to the internal CDC FIFO, ensuring data is committed only at a valid frame boundary. The block also accepts i_vcmp_enable and I_vcmp_done from the Valid Comparator to coordinate test-mode handshaking with the rest of the Valid Plane.

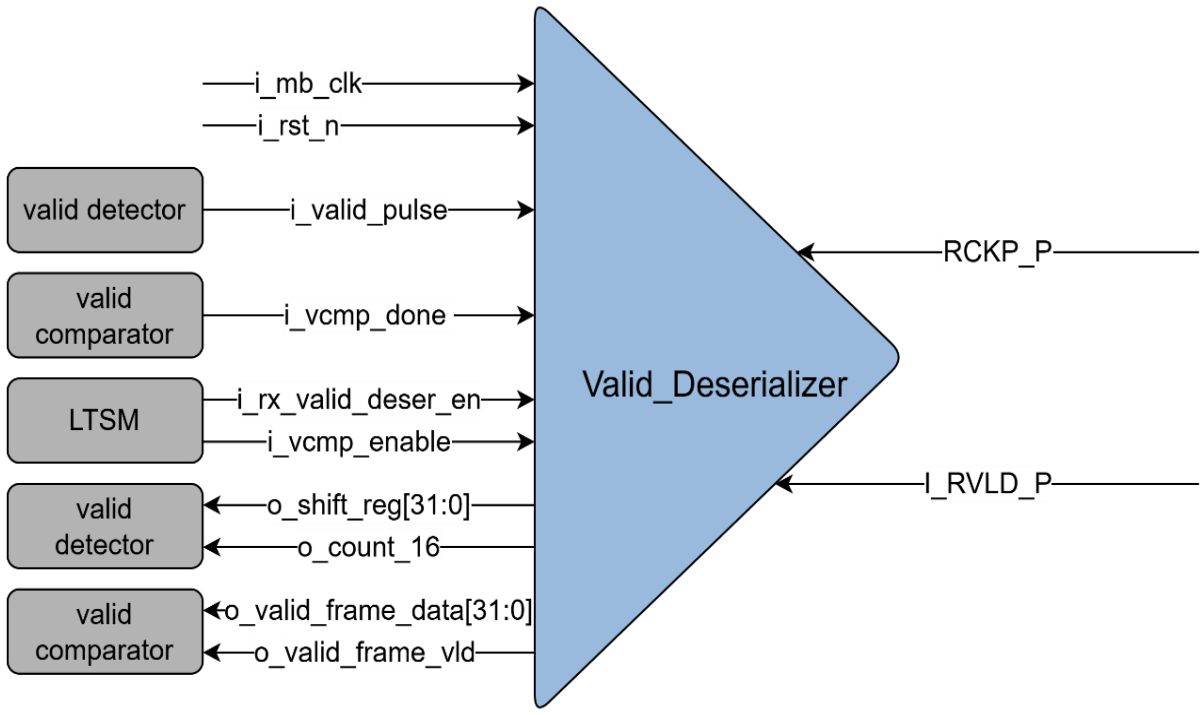


Figure 4-12: Valid Deserializer Block Diagram

Table 4-9: Valid Desierializer Interface signal

Signal Name	Direction	Width	Description
RCKP_P	Input	1	High-speed forwarded bit-rate clock used for DDR serial sampling.
i_mb_clk	Input	1	Recovered parallel digital clock ({PLL}/16) driving the backend logic.
i_rst_n	Input	1	Active-low asynchronous system reset.
i_rx_valid_deser_en	Input	1	Subsystem enable signal from internal control logic to activate the deserializer.
ser_data_in	Input	1	High-speed incoming serial bit stream from the physical Valid lane (i_RVLD_P).
i_valid_pulse	Input	1	Write increment control signal; qualifies when a deserialized frame word should be written to FIFO.
i_vcmp_enable	Input	1	The valid comparator enable
I_vcmp_done	Input	1	The valid comparator done signal which tells that the test is complete
o_shift_reg	Output	32	Observability output exposing the continuous, free-running parallel shift register.
o_valid_frame_data	Output	32	Synchronized, fully deserialized 32-bit valid framing data available to the parallel backend.
o_valid_frame_vld	Output	1	Read domain valid flag; asserts for 1 cycle when new data is updated on o_valid_frame_data.
o_count_16	Output	1	Terminal count indicator; asserts high when the frame loop counter reaches its terminal index 15.

4.3.2 Date Lane Deserializer

The unit_data_deserializer is a DDR deserializer for a single data lane in the UCIe Main-Band RX physical layer. It continuously samples the high-speed serial input (ser_data_in / i_RD_P[i]) on the RCKP_P clock domain. Frame-aligned capture is triggered by i_valid_frame_pulse from the Valid Plane, upon which the aligned 32-bit word is transferred to the i_mb_clk domain. The parallel output is available on o_par_data, with o_data_valid asserting for one cycle when new data is ready. Activation is controlled by i_rx_data_deser_en; reset by i_rst_n.

Like the Valid Deserializer, data is sampled on both edges of RCKP_P, building the 32-bit word two bits at a time. The data lane does not independently detect frame boundaries — it waits passively for i_valid_frame_pulse from the Valid Plane to trigger the FIFO write, guaranteeing that all 16 data deserializer instances commit and present their respective words at the exact same frame reference point.

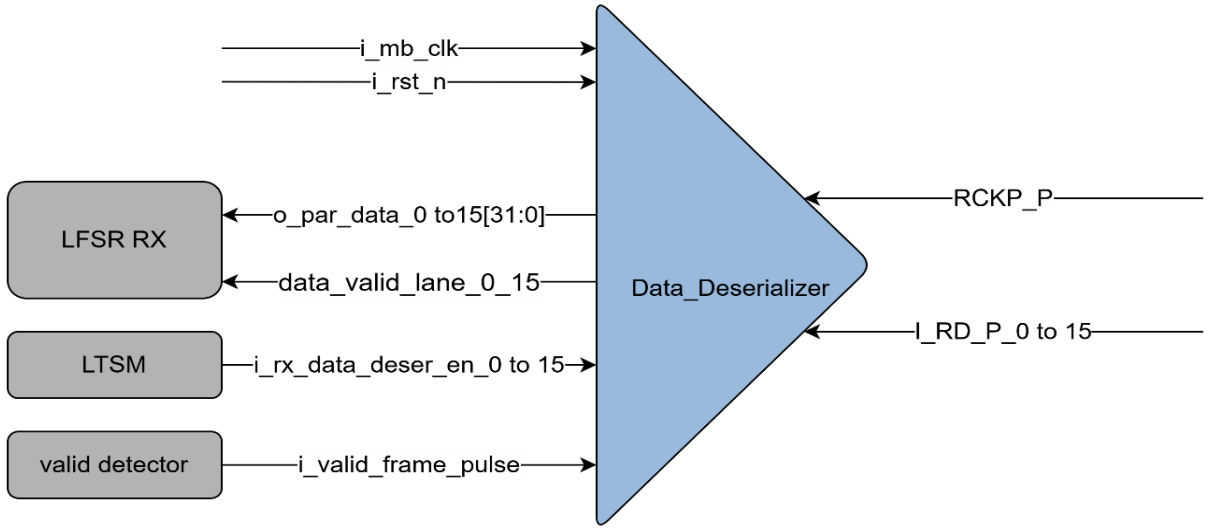


Figure 4-13: Data Deserializer

Table 4-10: Data Desierializer Interface signal

Signal Name	Direction	Width	Description
i_mb_clk	Input	1	Recovered parallel digital clock ({PLL}/16) driving the backend data processing logic.
RCKP_P	Input	1	High-speed forwarded bit-rate clock used for DDR serial sampling.
i_rst_n	Input	1	Active-low asynchronous system reset.
i_rx_data_deser_en	Input	1	Subsystem enable; when de-asserted, locks data sampling and flushes FIFO control states.
ser_data_in	Input	1	High-speed incoming serial bit stream from one of the physical data lanes (i_RD_P[i]).
i_valid_frame_pulse	Input	1	Gating frame-start pulse from the valid detector; triggers a write to the FIFO at the deterministic frame boundary.
o_par_data	Output	32	Fully deserialized and domain-synchronized 32-bit parallel lane word forwarded to the descrambler stage.
o_data_valid	Output	1	Read domain valid flag; asserts high for 1 cycle when a valid parallel word is available on o_par_data.

4.3.3 Valid Comparator

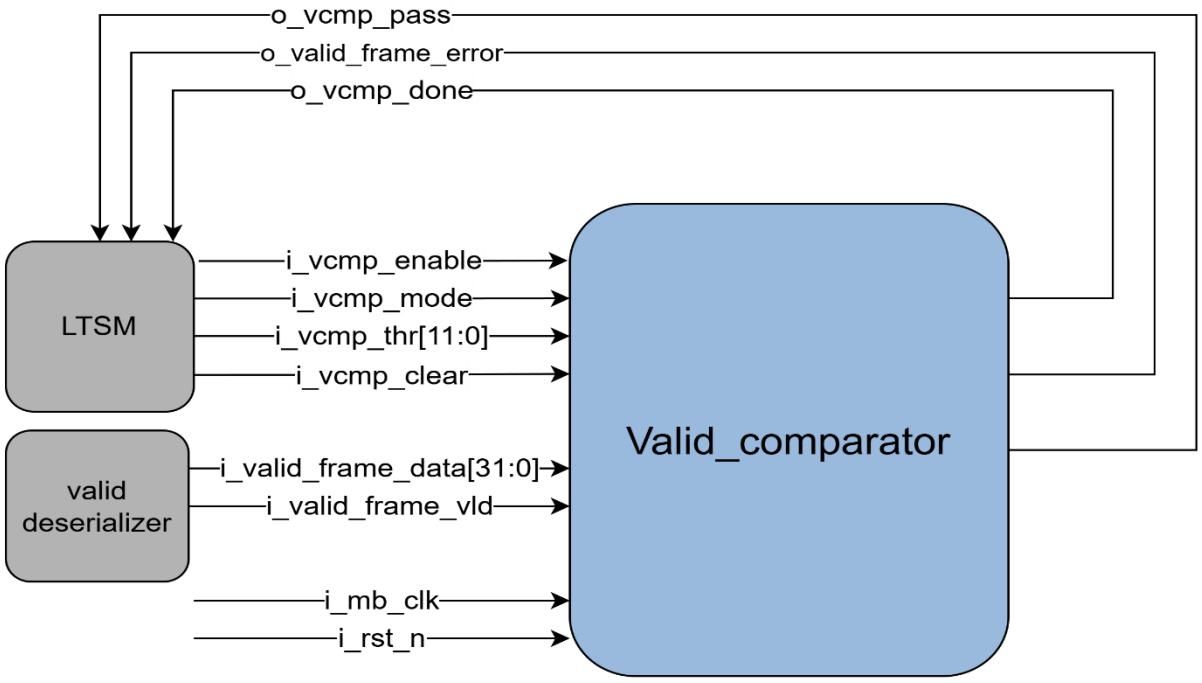


Figure 4-14: Valid Comparator

The unit_valid_comparator is a hardware verification engine for the UCIe Main-Band RX Valid Lane. It compares incoming deserialized framing data (i_valid_frame_data / i_valid_frame_vld) against the expected pattern (0X·F0F0F0F) over a fixed 128-byte test window. Two evaluation modes are selected by i_vcmp_mode: consecutive-match mode and bit-error threshold mode (threshold set by i_vcmp_thr). Testing is enabled by i_vcmp_enable and cleared by i_vcmp_clear. Results are reported on o_vcmp_done and o_vcmp_pass. Anomalous lane transitions during idle are flagged on o_valid_frame_error.

In consecutive-match mode, the incoming 32-bit word is sliced into bytes; a byte-level match streak must reach 16 consecutive correct bytes to assert o_vcmp_pass, and any mismatch immediately resets the streak. In bit-error threshold mode, all bit errors over the 128-byte window are accumulated and compared to i_vcmp_thr at completion. Outside active test windows, the block continuously monitors i_valid_frame_vld for unexpected lane transitions and emits a single-cycle pulse on o_valid_frame_error if any are detected.

Table 4-11: Valid Comparator Interface signal

Signal Name	Direction	Width	Description
i_mb_clk	Input	1	Recovered digital parallel clock domain ({PLL}/16) driving the tracking logic (MB_clk).
i_rst_n	Input	1	Active-low asynchronous system reset.
i_vcmp_enable	Input	1	Test execution control; drives the internal FSM into the active comparison window when high.
i_vcmp_mode	Input	1	Comparison algorithm selector: 0 for Consecutive Streak Mode, 1 for Bit-Error Threshold Mode.
i_vcmp_thr	Input	12	Maximum allowable bit-error tolerance value utilized during Threshold Mode evaluation.
i_vcmp_clear	Input	1	Synchronous clearing signal; forces FSM to IDLE and completely flushes error accumulators.
i_valid_frame_data	Input	32	Deserialized 32-bit parallel data word originating from the Valid Lane's CDC FIFO.
i_valid_frame_vld	Input	1	Data valid qualification signal indicating a newly assembled 32-bit word is ready for evaluation.
o_vcmp_done	Output	1	Status indicator; pulses or stays high to indicate that the fixed 128-iteration test sequence is finished.
o_vcmp_pass	Output	1	Binary verification flag indicating that the Valid Lane passed the active qualification test constraints.
o_valid_frame_error	Output	1	Status monitor flag; fires a single-cycle error pulse if illegal lane transitions appear during link idle states.

4.3.4 Clock Pattern Detector

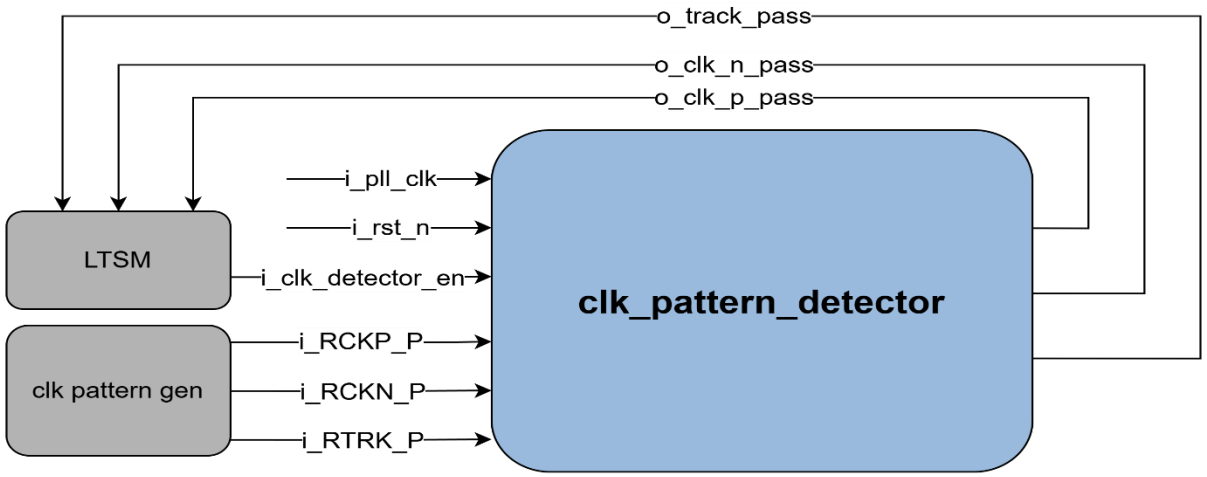


Figure 4-15: CLK Pattern Detector

The unit_clk_pattern_detector_rx validates clock training signals during UCIe Main-Band early link training states. When enabled by i_clk_detector_en, it samples i_RCKP_P, i_RCKN_P, and i_RTRK_P against the local i_pll_clk to verify the expected alternating toggle/quiet structural pattern. Each lane is independently tracked; once 16 consecutive valid pattern cycles are detected, the corresponding sticky pass flag (o_clk_p_pass, o_clk_n_pass, o_track_pass) is permanently asserted.

The expected training pattern consists of (16 clock cycles followed by 8 cycles of low). The positive clock component (i_RCKP_P), the negative component (i_RCKN_P), and the tracking lane (i_RTRK_P) are each checked independently. All three pass flags (o_clk_p_pass, o_clk_n_pass, o_track_pass) are sticky: once asserted, they remain high until the block is reset via i_rst_n, signaling that the clock infrastructure is ready for normal operation.

Table 4-12: Clk Pattern Detector Interface signals

Signal Name	Direction	Width	Description
i_pll_clk	Input	1	RX-local high-speed PLL verification clock utilized to sample training clock patterns.
i_rst_n	Input	1	Active-low asynchronous system reset.
i_clk_detector_en	Input	1	Global enablement control signal from the LTSM; activates pattern logging blocks when high.
i_RCKP_P	Input	1	Positive component of the incoming forwarded high-speed differential training clock.
i_RCKN_P	Input	1	Negative component of the incoming forwarded high-speed differential training clock.
i_RTRK_P	Input	1	Serial input stream originating from the physical high-speed clock tracking lane.
o_clk_p_pass	Output	1	Sticky qualification flag; asserts high when RCKP_P successfully completes 16 consecutive valid pattern cycles..
o_clk_n_pass	Output	1	Sticky qualification flag; asserts high when RCKN_P successfully completes 16 consecutive valid pattern cycles.
o_track_pass	Output	1	Sticky qualification flag; asserts high when RTRK_P successfully completes 16 consecutive valid pattern cycles.

4.3.5 LFSR RX

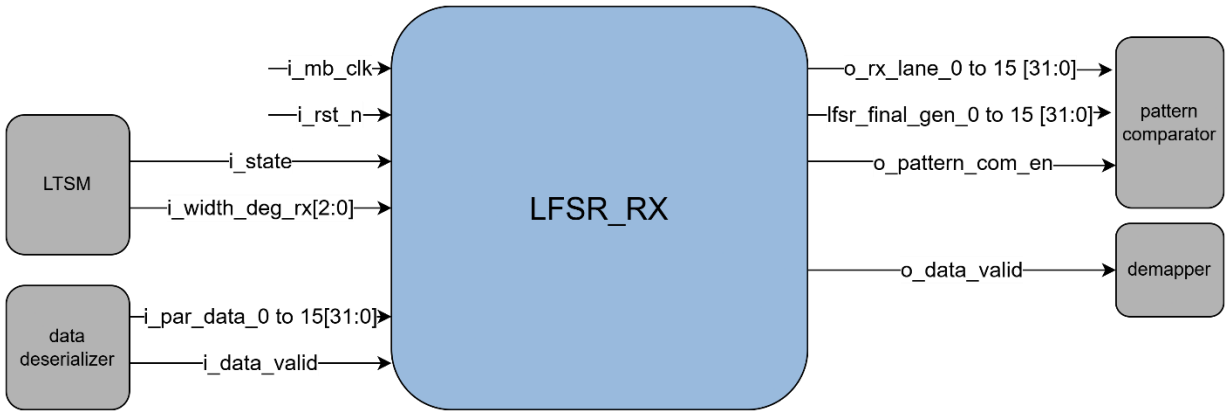


Figure 4-16: LFSR Rx

unit_lfsr_rx is a critical block within the receiver datapath of MB_RX_TOP. It performs the exact inverse operations of the transmitter's LFSR, implementing the UCIe PRBS-23 polynomial (G(X) = $x^{23} + x^{21} + x^{16} + x^{8} + x^{5} + x^{2} + 1$) across all active data lanes. To match the 32-bit throughput of the upstream deserializer array without introducing multi-cycle pipeline latency, the serial recurrence relation is combinationally unrolled by 32 steps, computing the full descramble or reference word in a single i_mb_clk cycle. The active operating mode is selected by i_state from the LTSM, and link-width adaptation across x16, x8, and x4 lane configurations is managed via i_width_deg_rx. The module unifies two primary operating modes:

Training Mode: It locally generates the reference PRBS-23 and Per-Lane ID (PLID) sequences, routing them to the PATTERN_COMPARATOR block for link quality verification, lane alignment, and bit-error-rate (BER) testing.

Active (Data) Mode: It descrambles the incoming raw payload data from the DATA_DES (Deserializer) block by executing a bitwise XOR with the locally synced LFSR state, feeding the descrambled payload directly to the DEMAPPER block.

The block contains eight physical LFSR instances, each mirrored across two lane slots to cover all 16 lanes under full-width x16 operation. Each time i_state transitions into a training phase, all instances are reloaded from fixed seed values, guaranteeing immediate bitwise phase alignment with the transmitter without requiring any out-of-band synchronization handshake.

In PRBS training states, o_final_gene carries the local gold reference and o_pattern_comp_en is asserted to activate the PATTERN_COMPARATOR, while o_rx_lane forwards the raw received data for direct comparison. In Per-Lane ID states, 16 unique lane tokens (formatted as 1010_<lane_index>_1010) are driven onto o_final_gene, enabling lane-swap detection and crossbar mapping resolution. In DATA_TRANSFER, each lane's i_par_data word is XOR-descrambled against the synchronized PRBS-23 output and presented on o_rx_lane with o_data_valid asserted. The i_width_deg_rx input maps the active LFSR engines to the correct lane subset for x16, x8, or

x4 operation, gating unused lane outputs to zero to maintain alignment with the transmitter's lane configuration.

Table 4-13: LFSR Rx Interface Signal

Signal Name	Direction	Width	Description
i_mb_clk	Input	1	Main receiver parallel clock for the Main Band RX path.
i_rst_n	Input	1	Asynchronous active-low global reset signal.
i_state	Input	3	Current Link Training State Machine (LTSM) state code.
i_width_deg_rx	Input	3	Active lane configuration/degradation status code.
i_data_valid	Input	1	Control gate from the RX buffer stage enabling LFSR stepping and active data capture
i_par_data	Input	32	16-channel array of parallel raw 32-bit input data words from the Deserializer (DATA_DES)
o_rx_lane	Output	32	Data output bus array. Carries raw data in training modes and fully descrambled payload words to the DEMAPPER during DATA_TRANSFER
o_final_gene	Output	32	Locally generated reference data array (PRBS32 or PLIDs) sent directly to the PATTERN_COMPARATOR.
o_pattern_comp_en	Output	1	Master enable switch for the pattern comparator; high during PATTERN_LFSR and PER_LANE_IDE phases.
o_data_valid	Output	1	Data validation flag; asserted exclusively during active DATA_TRANSFER cycles to validate descrambler output.

4.3.6 Pattern Comparator

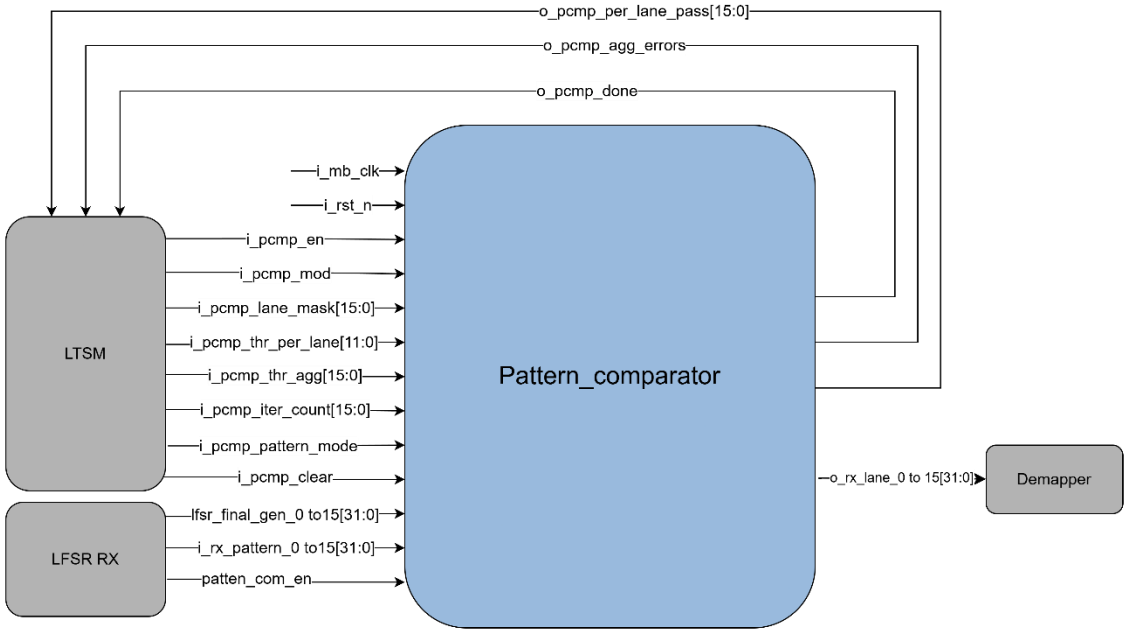


Figure 4-17: Pattern Comparator

The unit_mb_pattern_comparator is a multi-lane link quality verification block for the UCIe Main-Band RX datapath. During link training, it cross-compares the received data (i_rx_pattern) against the local LFSR/PLID reference (lfsr_final_gene) cycle-by-cycle over a configurable test window set by i_pcmp_iter_count, with the run started by I_pcmp_enable. Individual lanes can be selectively excluded from evaluation using i_pcmp_lane_mask, and the block is cleared at any time by i_pcmp_clear.

Two evaluation modes are available via i_pcmp_mode. In Per-Lane mode (i_pcmp_mode = 0), each lane is independently evaluated: the test pattern — either LFSR PRBS-23 or Per-Lane ID tokens — is selected by i_pcmp_pattern_mode, and per-lane pass/fail results are reported in o_pcmp_per_lane_pass against the threshold i_pcmp_thr_per_lane (sticky-fail in LFSR mode, sticky-pass in PLID mode). In Aggregate mode (i_pcmp_mode = 1), UI-level errors are accumulated link-wide and compared against i_pcmp_thr_aggregate, with the result reported on o_pcmp_agg_error. Test completion is signaled by o_pcmp_done.

Table 4-14: Pattern Comparator Interface signal

Signal Name	Direction	Width	Description
i_mb_clk	Input	1	Parallel processing clock for the Main Band RX datapath (mb_clk).
i_rst_n	Input	1	Global asynchronous active-low system reset.
I_pcmp_enable	Input	1	Master test control line; driving this line high resets metrics and starts a comparison run from S_IDLE
i_pcmp_mode	Input	1	Selects comparison tracking layer: 0 = Per-Lane tracking, 1 = Aggregate module tracking.
i_pcmp_lane_mask	Input	16	Lane exclusion mask bitmask. Active-high (1) drops the target lane out of evaluations
i_pcmp_thr_per_lane	Input	12	The allowable bit-error ceiling for an individual lane during an LFSR test sequence.
i_pcmp_thr_aggregate	Input	16	The allowable global UI-error count ceiling for the entire module during aggregate testing.
i_pcmp_iter_count	Input	16	Defines the precise runtime length of the active comparison test window (in clock cycles).
i_pcmp_pattern_mode	Input	1	Specifies test pattern style: 0 = LFSR PRBS-23 pattern, 1 = Per-Lane ID token mode.
i_pcmp_clear	Input	1	Synchronous register flush control line; overrides active state and forces block back to default S_IDLE settings.
lfsr_final_gene	Input	32	16-channel array of parallel expected reference gold-data words from the local LFSR block (o_final_gene).
i_rx_pattern	Input	32	16-channel array of incoming raw parallel data words from the link partner (o_Data_by).
i_pattern_comp_en	Input	1	Secondary validation gate input from the LFSR block (pattern_comp_en); must be high to cycle error counters.
o_pcmp_done	Output	1	Test completion flag; goes high when the iteration threshold is achieved, validating outputs.
o_pcmp_per_lane_pass	Output	16	Multi-lane status bus. Latches low (0) on error threshold overrun in LFSR mode; latches high (1) on clear sequence detection in PLID mode.
o_pcmp_agg_error	Output	1	System-level failure indicator; driven high if the overall module error count surpasses the aggregate allowance limit.

4.3.7 Demapper

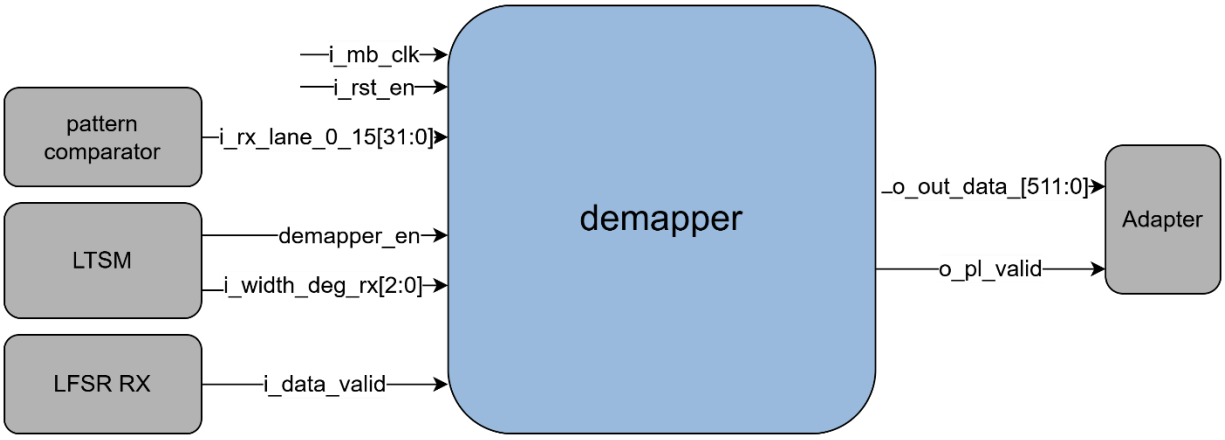


Figure 4-18: Demapper

The unit_demapper reverses the transmitter mapper operation, merging 16 parallel 32-bit lane streams (i_rx_lane_0 to i_rx_lane_15) into a unified 512-bit output word (o_out_data) by reassembling data byte-by-byte across lanes. The accumulation cycle count adapts to the active link width via i_width_deg_rx: 1 cycle for x16 full-width, 2 cycles for x8 half-width, and 4 cycles for x4 quarter-width mode. o_pl_valid is asserted when the output word is complete. Operation is gated by demapper_en and i_data_valid; reset by i_rst_n.

Data is re-ordered byte-by-byte across the lanes in descending bit-field order ([31:24], [23:16], [15:8], [7:0]) to reverse the transmitter's interleaving scheme and reconstruct the original continuous payload bitstream. When fewer than 16 lanes are active, the demapper accumulates partial words across multiple i_mb_clk cycles — 2 cycles for x8, 4 cycles for x4 — before asserting o_pl_valid, maintaining a consistent 512-bit output word to the upper protocol layer regardless of the active lane width.

Table 4-15: Demapper Interface signals

Signal Name	Direction	Width	Description
i_mb_clk	Input	1	Main parallel digital clock for backend data processing (mb_clk domain).
i_rst_n	Input	1	Global async/sync reset line
i_rx_lane_0 to i_rx_lane_15	Input	32 each	16 individual parallel 32-bit lane data streams from the upstream LFSR_RX descrambler.
demapper_en	Input	1	Master block enable signal
i_data_valid	Input	1	Validation flag indicating valid data present on inputs
i_width_deg_rx	Input	3	Dynamic lane descrambler configuration mode
o_pl_valid	Output	1	Parallel layer strobe/go signal indicating valid payload
o_out_data	Output	512 bits	Complete reassembled wide payload word ready for upper layers

5.0 Sideband Design

5.1 Subsystem Overview

SideBand_Top is the top-level integration module of the sideband subsystem. Its purpose is to transport the three logical message categories defined by the UCIe specification — training and initialization coordination messages, RDI link-management messages, and register-access transactions — across the low-speed forwarded-clock serial interface that connects the two communicating dies. The subsystem operates on a permanently powered auxiliary clock domain, ensuring that sideband communication remains functional across all mainband power states, including states in which the mainband PLL is disabled or the gated mainband clock is suppressed.

The subsystem is bounded on its lower side by the serial pins RXCKSB, TXCKSB, RXDATASB, and TXDATASB, which connect to the analog hard macro. On its upper side it presents four named interfaces: an LTSM message channel to MainSM, an RDI message channel to the RDI state machine within MainSM, a configuration transaction channel to the D2D Adapter, and a register-access bus to Reg_File. SideBand_Top also exports the derived clk_sb signal, which is the sole clock of the Reg_File subsystem.

All behavioral logic within SideBand_Top is partitioned into four digital sub-blocks — Link_Controller, Training_Mgmt, RDI_control, and Reg_Access — plus the analog hard macro that provides the SerDes, clock generation, and serial-pin drivers.

5.1.1 SideBand Internal Module Hierarchy

SideBand_Top instantiates five components. The analog hard macro (sideband analog hard macro) owns the 800 MHz PLL, the divide-by-eight clock divider that produces clk_sb, the 8:1 serializer, and the 1:8 deserializer, together with the serial pin drivers and receivers for TXDATASB, TXCKSB, RXDATASB, and RXCKSB. All signals that cross the digital-to-analog boundary travel through this component; no direct connection exists between the four digital sub-blocks and the serial pins.

The Link_Controller is the digital packet framing and pattern generation engine. It arbitrates outbound 128-bit messages from Training_Mgmt and RDI_control, breaks them into 64-bit serial words through sb_mapper, steers the serializer between pattern mode and message mode through sb_pattern_engine, and on the receive path passes incoming 64-bit words through sb_demapper and LINK_Demux to the appropriate consumer. Training_Mgmt manages the LTSM and RDI message paths: it bridges lclk-domain inputs from MainSM across asynchronous FIFOs into the clk_sb domain, arbitrates between the two transmit sources using a round-robin arbiter, and on the receive side decodes and routes inbound training packets back to MainSM. RDI_control manages the adapter configuration channel: it reassembles 32-bit lp_cfg chunks into 128-bit SB packets, routes them downstream to the link or locally to Reg_Access, and on the upstream path breaks outgoing packets into 32-bit pl_cfg chunks under credit-counter flow control. Reg_Access

bridges inbound register-access SB packets to the Reg_File bus, drives the address, byte-enable, and data signals, and generates UCIe-compliant completion packets in response.

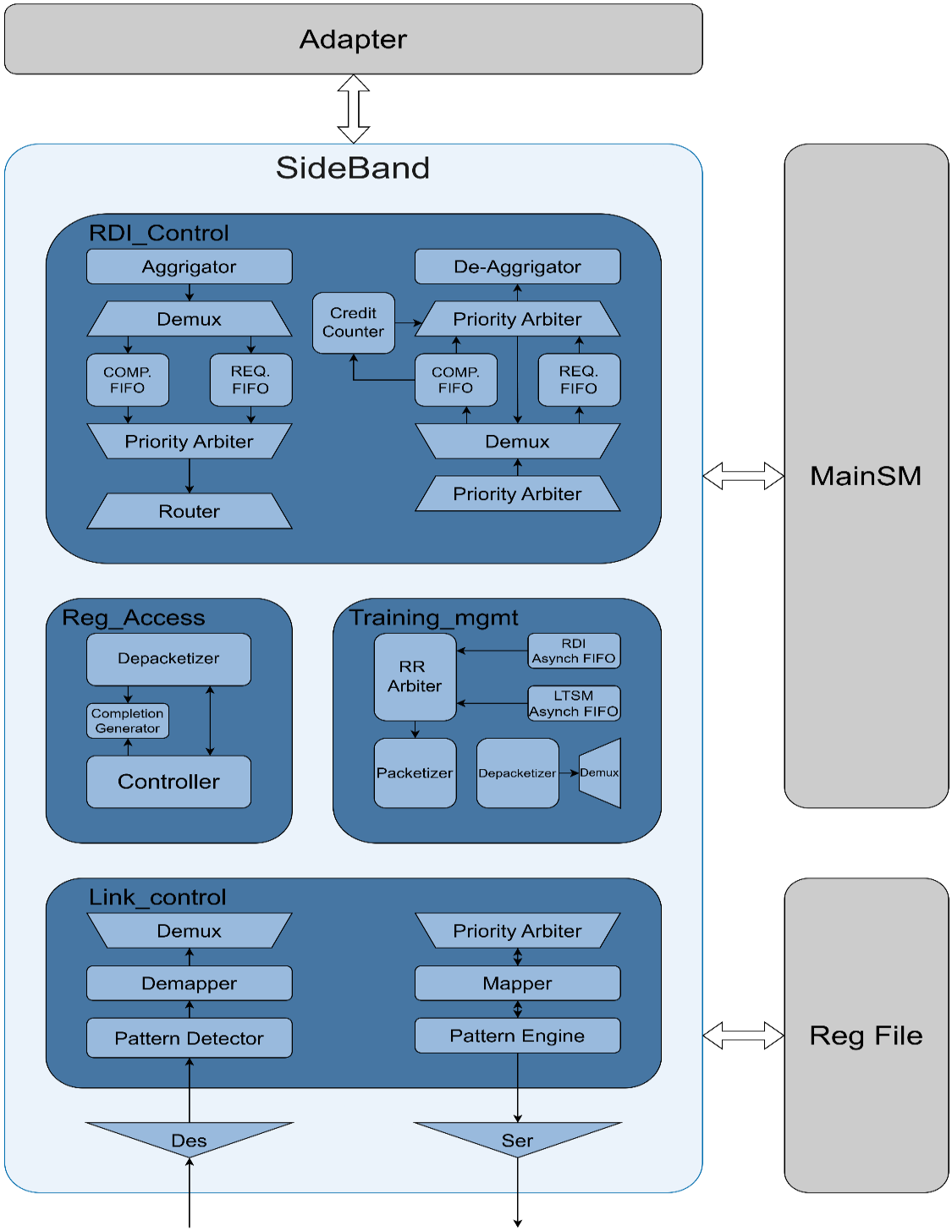


Figure 5-1: SideBand Hierarchy Architecture

5.2 Sideband SerDes

The Sideband SerDes comprises two complementary components — sb_serializer and sb_deserializer — that together implement the parallel-to-serial and serial-to-parallel conversion required on the sideband forwarded-clock interface. Both components are instantiated within the sideband_analog_hard_macro and operate across two clock domains: the 800 MHz serial clock (sb_pll_clock) and the 100 MHz parallel clock (clk_sb). The SerDes constitutes the lowest-level boundary between the digital packet processing logic and the physical serial pins of the sideband interface.

5.2.1 SB Serializer

The sb_serializer converts 64-bit parallel data words received from the Link_Controller into a bit-serial stream on TXDATASB, simultaneously forwarding the serialization clock on TXCKSB. The module implements a three-state FSM in the 800 MHz serial clock domain: S_IDLE, S_SHIFT, and S_GAP. On receiving a valid parallel word, the serializer loads it into an internal shift register and transitions to S_SHIFT, emitting one bit per serial clock cycle beginning with bit 0. After the 64th bit is transmitted, the serializer transitions to S_GAP and holds TXDATASB low for 32 serial clock cycles, enforcing the mandatory inter-packet gap required by the UCIe specification. When PMO is active, the S_GAP state is bypassed entirely and the next packet is loaded immediately at the completion of the preceding 64-bit sequence, reducing the per-packet interval from 96 UI to 64 UI. The forwarded clock TXCKSB is generated by a gated clock cell driven by the shift_active condition, ensuring TXCKSB is present only during the S_SHIFT phase.

A skid buffer manages the parallel-domain to serial-domain crossing: the parallel data word is captured in the clk_sb domain and a toggle-based CDC mechanism notifies the serial domain when new data is available. The tx_rdy signal is deasserted while the buffer holds an unconsumed word and reasserted once the serial domain has consumed it, preventing overwriting of in-transit data.

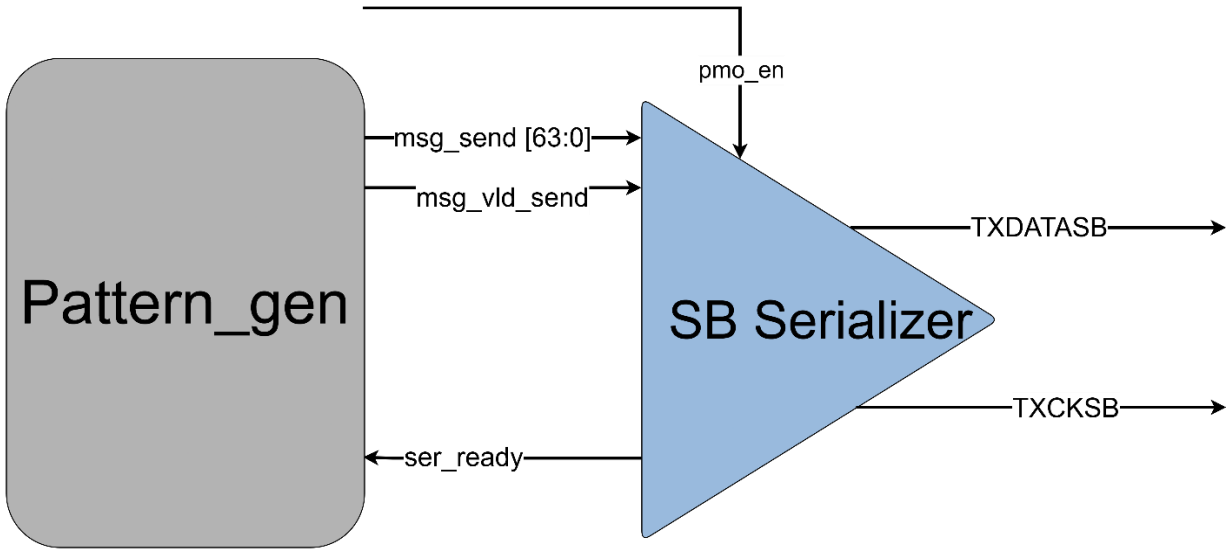


Figure 5-2: SB Serializer

Table 5-1: SB serializer Interface Signals

Signal	Direction	Width	Description
clk_parallel	Input	1	Parallel-domain clock (clk_sb, 100 MHz)
clk_serial	Input	1	Serial-domain clock (sb_pll_clock, 800 MHz)
rst_n	Input	1	Active-low reset
pmo_en	Input	1	PMO enable; bypasses S_GAP state for back-to-back packet transmission
tx_parallel_data	Input	64	64-bit parallel data word to serialize
tx_data_valid	Input	1	Parallel data valid; initiates serialization when tx_rdy is asserted
tx_rdy	Output	1	Serializer ready to accept a new parallel word
TXDATASB	Output	1	Serialized sideband data output (LSB first)
TXCKSB	Output	1	Forwarded sideband clock; active only during S_SHIFT state

5.2.2 PMO Enable Path

The PMO enable path governs whether the serializer operates in standard or Performant Mode. A combinational assignment in SideBand_Top derives the effective serializer PMO enable signal (ser_pmo_en) as the logical AND of pmo_en with the complement of pattern_mode. This ensures that during SBINIT pattern transmission, the serializer always operates in standard mode regardless of the negotiated PMO state, because inter-packet spacing is required for pattern integrity during initialization. Once pattern_mode is deasserted and the link has transitioned to message mode with pmo_en asserted, ser_pmo_en becomes active and the serializer begins issuing back-to-back 64-bit packets without inter-packet gaps. The pmo_en input to SideBand_Top is driven by MainSM following successful negotiation of Sideband Performant Mode Operation during the MBINIT.PARAM sub-state exchange.

5.2.3 SB De-serializer

The sb_deserializer reconstructs a 64-bit parallel word from the incoming bit-serial stream on RXDATASB, using the received forwarded clock RXCKSB as the sampling reference. The module maintains a 64-bit shift register clocked by RXCKSB. On each rising edge of RXCKSB, the incoming

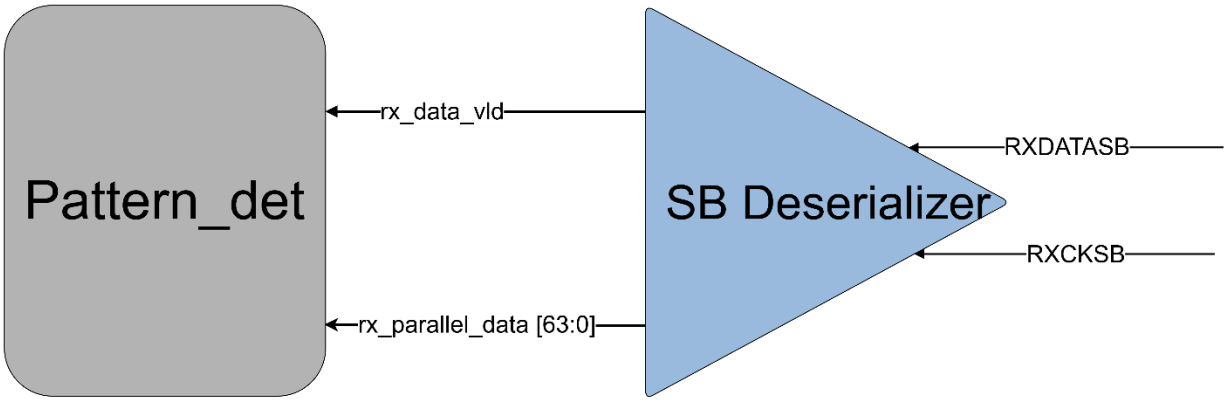


Figure 5-3: SB Deserializer

RXDATASB bit is inserted into the most significant bit position while the remaining bits shift right, implementing LSB-first reception consistent with the serializer's LSB-first transmission order.

A bit counter tracks the current position within the 64-bit packet; when the counter reaches 63, the shift register contents are captured into the parallel output register and a toggle signal is generated to communicate completion to the parallel clock domain.

Clock domain crossing from the RXCKSB domain to the clk_sb domain is implemented through a three-stage synchronizer chain on the toggle signal. A transition on the synchronized toggle causes the parallel output register contents to be presented on rx_parallel_data_out with rx_data_vld asserted for a single clk_sb cycle. This CDC mechanism ensures correct behavior across the frequency difference between the received forwarded clock and the local sideband parallel clock.

Table 5-2: SB Deserializer Interface signals

Signal	Direction	Width	Description
RXCKSB	Input	1	Received sideband forwarded clock from remote serializer
clk_parallel	Input	1	Parallel-domain clock (clk_sb, 100 MHz)
rst_n	Input	1	Active-low reset
RXDATASB	Input	1	Received sideband serial data (LSB first)
rx_parallel_data_out	Output	64	Deserialized 64-bit parallel data word (clk_sb domain)
rx_data_vld	Output	1	Single-cycle valid pulse in clk_sb domain when a 64-bit packet is complete

5.3 Link Controller

The Link_Controller is the central packet framing and traffic management block of the sideband subsystem. It performs two separable functions. In the transmit direction, it arbitrates between competing 128-bit message sources from Training_Mgmt and RDI_control, breaks each 128-bit message into two sequential 64-bit serial words through the sb_mapper, and selects between pattern mode and message mode through the sb_pattern_engine before presenting words to the serializer. In the receive direction, it accepts 64-bit words from the deserializer, reassembles them into 128-bit messages through the sb_demapper, and routes reassembled messages to the appropriate destination through the LINK_Demux. All logic within Link_Controller operates in the clk_sb domain; clock domain crossings for the LTSM and RDI paths are handled entirely within Training_Mgmt.

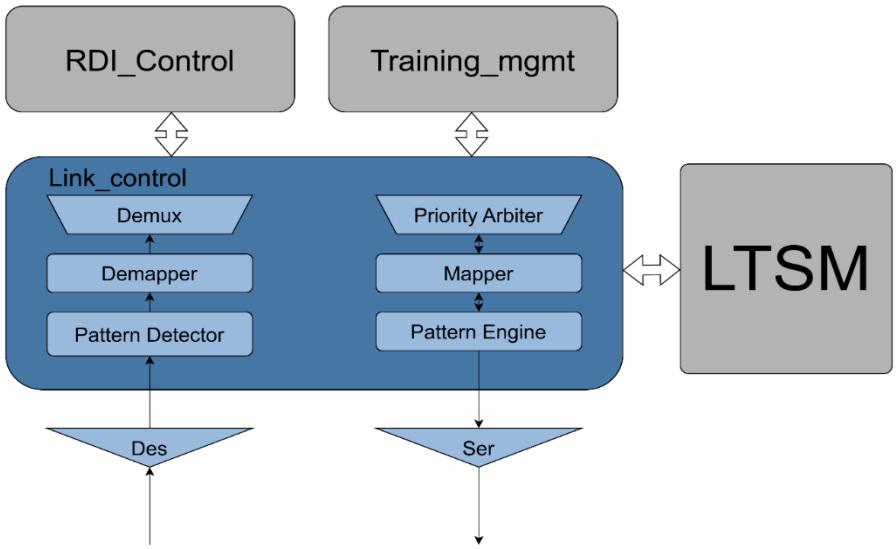


Figure 5-4: Link Control Diagram

5.3.1 TX Priority Arbiter

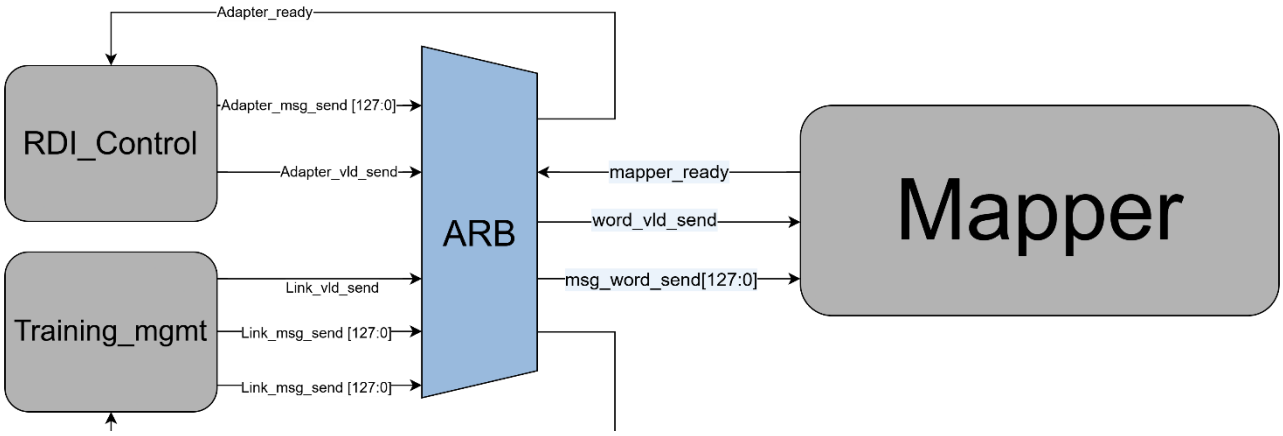


Figure 5-5: Tx Priority Arbiter

The TX priority arbiter (sb_priority_arbiter) within Link_Controller selects between the two competing 128-bit transmit sources: training messages from Training_Mgmt and adapter messages from RDI_control. The arbiter implements a fixed-priority scheme in which training messages are treated as the high-priority input and adapter messages as the low-priority input. When both inputs present valid data simultaneously, the training message is selected unconditionally and the adapter input is held until the high-priority transfer is complete. The selected 128-bit message is forwarded to sb_mapper for packetization into two sequential 64-bit serial words. Independent ready signals are returned to each source to indicate acceptance of the current message.

Table 5-3: TX Priority Arbiter Interface Singals

Signal	Direction	Width	Description
hip_msg	Input	128	High-priority message input (training messages from Training_Mgmt)
hip_vld	Input	1	High-priority message valid
hip_rdy	Output	1	High-priority source acceptance (ready) signal
lop_msg	Input	128	Low-priority message input (adapter messages from RDI_control)
lop_vld	Input	1	Low-priority message valid
lop_rdy	Output	1	Low-priority source acceptance (ready) signal
out_msg	Output	128	Arbitrated output message to sb_mapper
out_vld	Output	1	Arbitrated output valid
out_rdy	Input	1	Downstream (sb_mapper) ready

5.3.2 SB Mapper

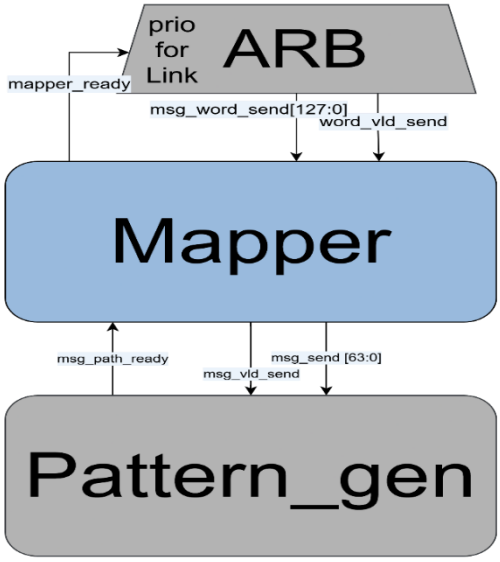


Figure 5-6: SB Mapper

The sb_mapper accepts 128-bit assembled message words from the TX arbiter and splits each word into two sequential 64-bit serial words for delivery to the serializer. The first 64-bit word carries the sideband packet header (bits 63:0 of the 128-bit message), and the second carries the data payload (bits 127:64). A valid-ready handshake governs the interface with the arbiter on the input side and with the serializer on the output side, ensuring both words of a message are delivered back-to-back without gap insertion between them. The mapper asserts mapper_rdy to the arbiter only when the serializer is ready and no previous transmission is in progress, preventing new message acceptance while an existing 128-bit message is being serialized.

Table 5-4: SB Mapper Interface signals

Signal	Direction	Width	Description
clk	Input	1	Sideband parallel clock (clk_sb)
rst_n	Input	1	Active-low reset
msg_word_send	Input	128	128-bit assembled message from TX priority arbiter
word_vld_send	Input	1	Message word valid
ser_rdy	Input	1	Serializer ready for next 64-bit word
mapper_rdy	Output	1	Mapper ready to accept a new 128-bit message
msg_send	Output	64	64-bit serializer word (header on first transfer, payload on second)
msg_vld_send	Output	1	Serializer word valid

5.3.3 SB Demapper

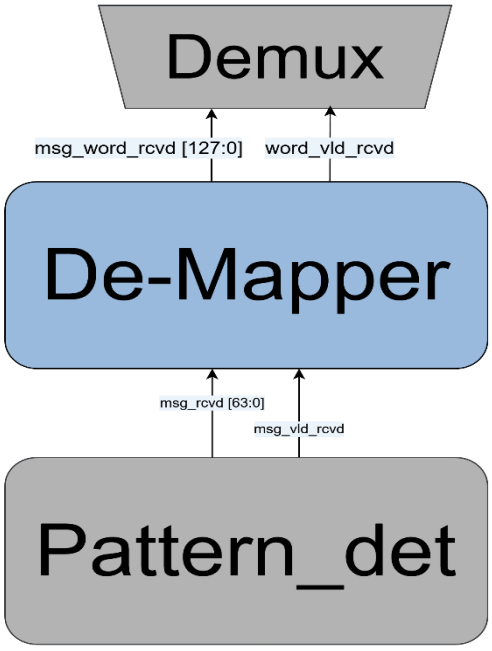


Figure 5-7: SB Demapper

The sb_demapper is the receive-side counterpart of sb_mapper. It accepts consecutive 64-bit words from the pattern detector and reassembles them into a 128-bit message word. The first received 64-bit word is captured as the header and the second as the payload, producing a complete 128-bit message that is presented to LINK_Demux with a single valid pulse. The demapper maintains a one-bit phase state to track whether the next incoming 64-bit word is a header or a payload, toggling between the two phases on each valid received word. This ensures correct message boundary alignment provided the transmitting mapper enforces atomic two-word sequences.

Table 5-5: SB Demapper Iterface signals

Signal	Direction	Width	Description
clk	Input	1	Sideband parallel clock (clk_sb)
rst_n	Input	1	Active-low reset
msg_rcvd	Input	64	64-bit received word from sb_pattern_detector pass-through
msg_vld_rcvd	Input	1	Received word valid
msg_word_rcvd	Output	128	Reassembled 128-bit message word
word_vld_rcvd	Output	1	128-bit message valid (asserted after second 64-bit word received)

5.3.4 Link Demux

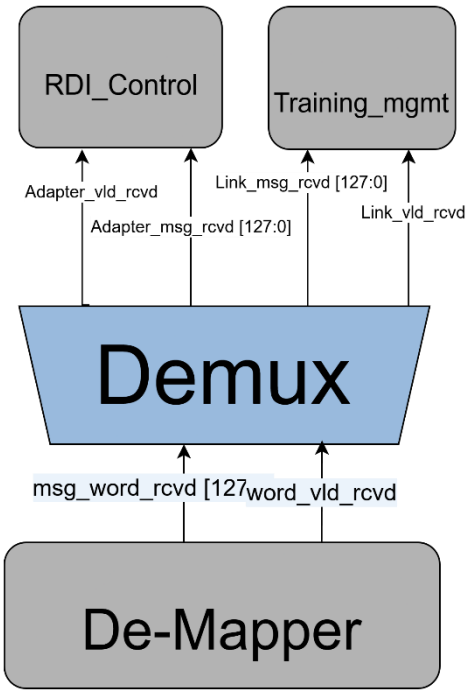


Figure 5-8: Link Demux

The LINK_Demux routes reassembled 128-bit messages from sb_demapper to the appropriate upper-layer consumer based on the message identifier fields embedded in the packet header. Messages belonging to the training and initialization coordination flow are routed to Training_Mgmt; messages associated with RDI adapter configuration or completion traffic are routed to RDI_control. Register-access packets flow through RDI_control rather than directly from LINK_Demux, as RDI_control is responsible for request-completion pairing, credit accounting, and local-versus-remote routing decisions. The LINK_Demux performs purely combinational steering based on opcode and source-destination identifier fields; it holds no state and introduces no latency beyond combinational propagation delay.

Table 5-6: Link Demux Interface signals

Signal	Direction	Width	Description
msg_word_rcvd	Input	128	Reassembled 128-bit message from sb_demapper
word_vld_rcvd	Input	1	128-bit message valid
adapter_msg_rcvd	Output	128	Message routed to RDI_control (adapter and register-access traffic)
adapter_vld_rcvd	Output	1	Adapter-routed message valid
trn_msg_rcvd	Output	128	Message routed to Training_Mgmt (LTSM and RDI training traffic)
trn_vld_rcvd	Output	1	Training-routed message valid

5.3.5 SB Pattern Generation and Detection Mode

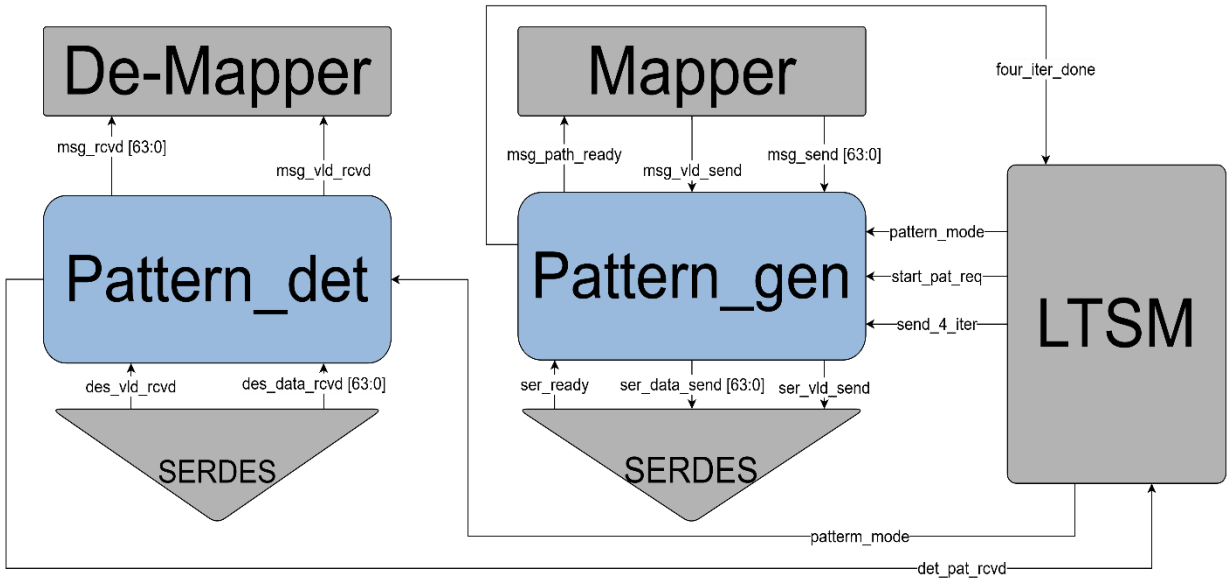


Figure 5-9: SB Pattern Generation and Detection

Two sub-blocks within Link_Controller implement the SBINIT sideband initialization pattern protocol. The sb_pattern_engine steers the serializer between SBINIT pattern mode and normal

message mode. When pattern_mode is asserted and start_pat_req is received, the engine continuously presents the 64-bit alternating clock pattern (0x5555_5555_5555_5555) to the serializer while blocking the sb_mapper output path. After the LTSM confirms that the remote die has detected the pattern, the engine continues transmitting for the number of additional iterations specified by req_iter_count and then asserts iter_done. When pattern_mode is deasserted, the engine reverts to pass-through, forwarding sb_mapper data to the serializer.

The sb_pattern_detector operates on the receive path. It monitors each 64-bit word from the deserializer and compares it against the clock pattern constant. An internal counter increments on each consecutive matching word; upon observing two consecutive matches — satisfying the 128 UI detection criterion defined by the specification — det_pat_rcvd is asserted to the LTSM for one clock cycle. During pattern mode, non-matching words are suppressed and not forwarded to sb_demapper. Once pattern_mode is deasserted, all received words are forwarded without filtering.

Table 5-7: SB Pattern Generation and Detection Mode Interface signals

Signal	Direction	Width	Description
clk	Input	1	Sideband parallel clock (clk_sb)
rst_n	Input	1	Active-low reset
pattern_mode	Input	1	Enable pattern generation mode
start_pat_req	Input	1	Request to begin pattern transmission
send_4_iter	Input	1	Detection confirmed; transmit remaining additional iterations
ser_rdy	Input	1	Serializer ready for next word
mapper_data	Input	64	Normal message data from sb_mapper (used in message mode)
mapper_valid	Input	1	sb_mapper output valid
mapper_rdy	Output	1	Ready acknowledgment back to sb_mapper (masked in pattern mode)
iter_done	Output	1	All additional pattern iterations transmitted
ser_data	Output	64	Selected 64-bit word to serializer (pattern constant or message)
ser_valid	Output	1	Serializer word valid

5.4 Training Management

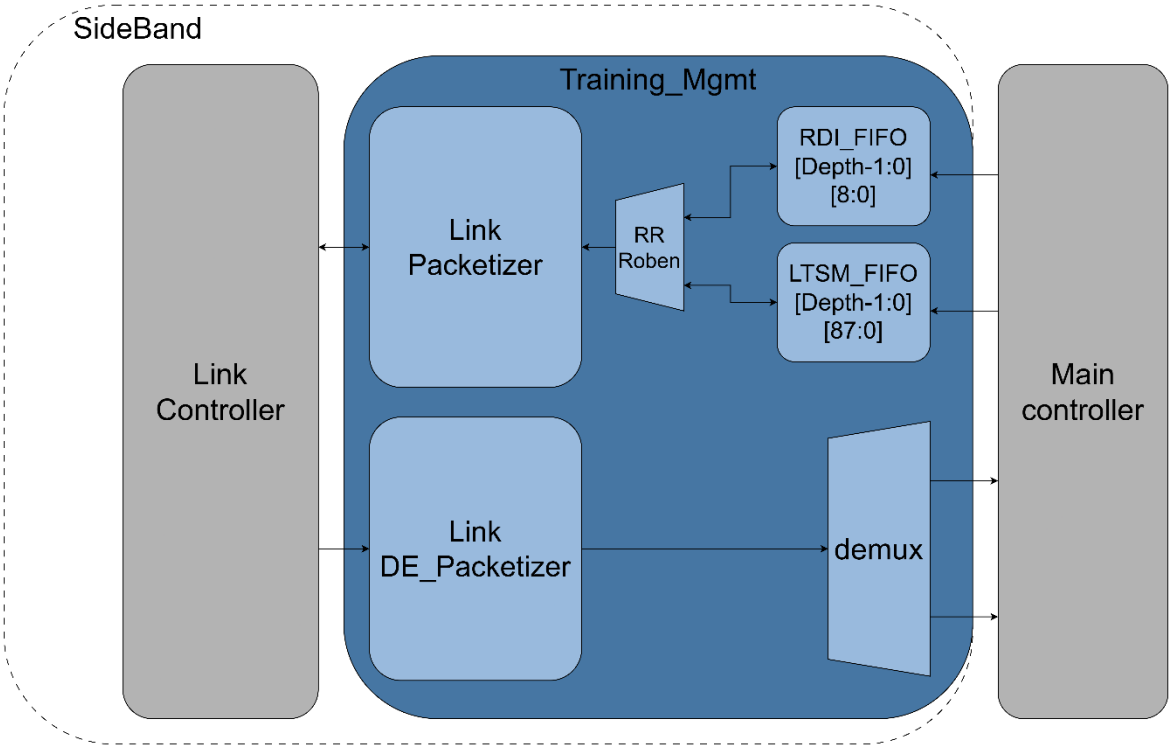


Figure 5-10: Train Management

Training_Mgmt is the sub-block responsible for bridging the LTSM and RDI state machine — which operate in the lclk domain within MainSM — to the clk_sb-domain packet processing infrastructure of the sideband subsystem. It manages two independent message paths: the LTSM training message path, which conveys training -phase sideband protocol messages identified by a message number and optionally carrying MsgInfo and data fields, and the RDI management message path, which conveys link state coordination messages between the RDI_SM and the remote die.

On the transmit side, Training_Mgmt accepts messages from MainSM into asynchronous FIFOs that bridge the clock domain boundary, then arbitrates between the two outgoing paths and packetizes the selected message into a 128-bit SB packet for delivery to Link_Controller. On the receive side, it accepts inbound 128-bit packets from Link_Controller and routes them to the correct upper-layer consumer based on the decoded message type.

5.4.1 TX Packetization

The Packetizer sub-block within Training_Mgmt converts decoded message fields — message number, MsgInfo, and data — into a 128-bit sideband packet conforming to the UCIe packet structure. The message number is used to index into the message codec package (msg_codec_pkg) to recover the opcode, source identifier, and destination identifier required to populate the 64-bit header. The MsgInfo and data fields are placed in the corresponding header extension and payload positions. The completed 128-bit packet is then presented to the RR_arbiter output interface and

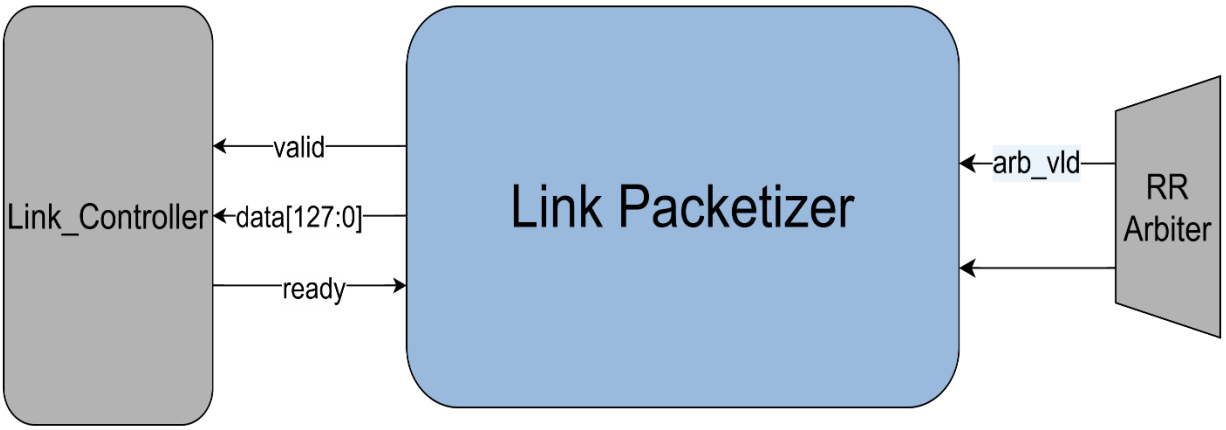


Figure 5-11: TX Packetizer Block Diagram

subsequently forwarded to Link_Controller. The Packetizer operates entirely in the clk_sb domain, receiving its inputs from the read ports of the asynchronous FIFOs rather than directly from MainSM.

Table 5-8: TX Packetization Interface signals

Signal	Direction	Width	Description
clk	Input	1	Sideband parallel clock (clk_sb)
rst_n	Input	1	Active-low reset
msg_data_send	Input	64	Message data field from RR_arbiter output
msg_info_send	Input	16	MsgInfo field from RR_arbiter output
msg_no_send	Input	8	Message number (index into msg_codec_pkg)
stall_send	Input	1	RDI stall flag to encode into packet header
valid_send	Input	1	Input valid from RR_arbiter
trn_rdy	Input	1	Link_Controller TX backpressure
trn_msg_send	Output	128	128-bit assembled SB packet
trn_vld_send	Output	1	Packet valid
rdy	Output	1	Packetizer ready (backpressure return to RR_arbiter)

5.4.2 Link DePacketizer

The DePacketizer sub-block within Training_Mgmt decodes inbound 128-bit packets received from Link_Controller. It extracts the opcode from the header, identifies the message type, and recovers the MsgInfo and data payload fields. A stall flag is decoded from the packet when the message type corresponds to an RDI link management operation. The decoded fields are presented to the Training_Mgmt_Demux for routing to the appropriate upper-layer consumer.

The DePacketizer performs header interpretation and field extraction only; it does not modify packet content or inspect payload semantics.

Table 5-9: Link DePacketizer Interface signals

Signal	Direction	Width	Description
clk	Input	1	Sideband parallel clock (clk_sb)
rst_n	Input	1	Active-low reset
msg_in	Input	128	128-bit received SB packet from Link_Controller
vld_in	Input	1	Packet valid
msg_no_out	Output	8	Decoded message number
msginfo_r	Output	16	Decoded MsgInfo field
payload_r	Output	64	Decoded data payload
vld_r	Output	1	Decoded fields valid
stall_rcvd	Output	1	Decoded RDI stall flag

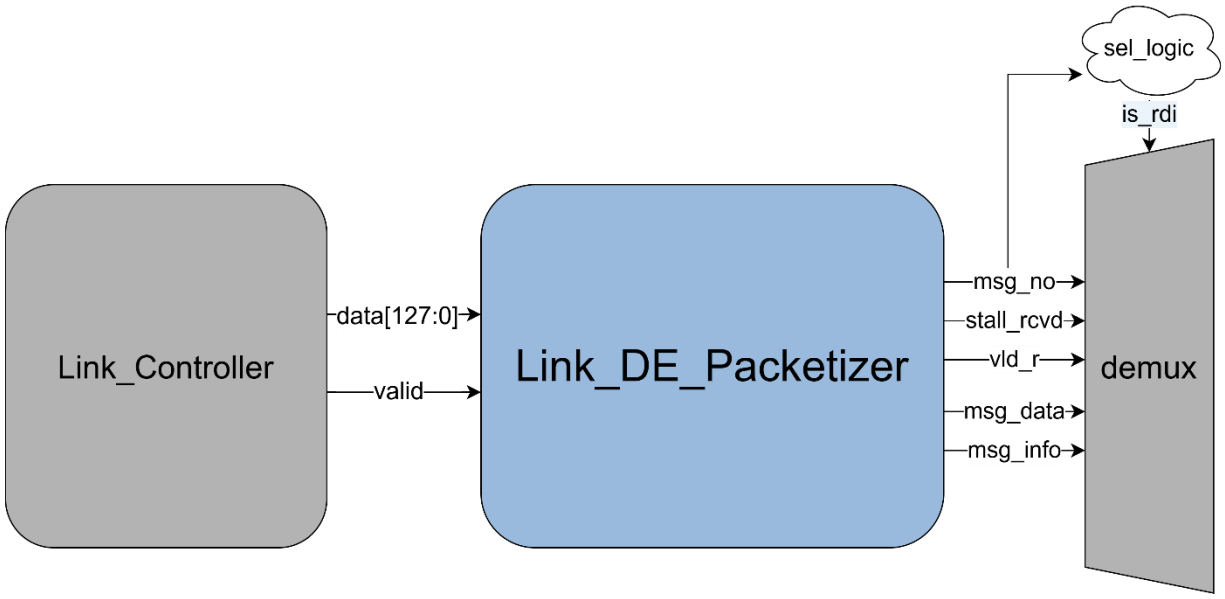


Figure 5-12: Link DePacketizer Block Diagram

5.4.3 RR Arbiter

The round-robin arbiter (RR_arbiter) within Training_Mgmt selects between the LTSM transmit FIFO and the RDI transmit FIFO when both have pending messages in the clk_sb domain. A round-robin scheduling policy ensures that neither source is indefinitely starved when both are simultaneously active, which can occur during the MBINIT phase when the LTSM sends parameter exchange messages while the RDI_SM simultaneously issues link management notifications. The arbitrated output is a packed 88-bit word carrying the message number, MsgInfo, data, and stall fields from the winning source. This packed representation is forwarded to the Packetizer for SB packet assembly.

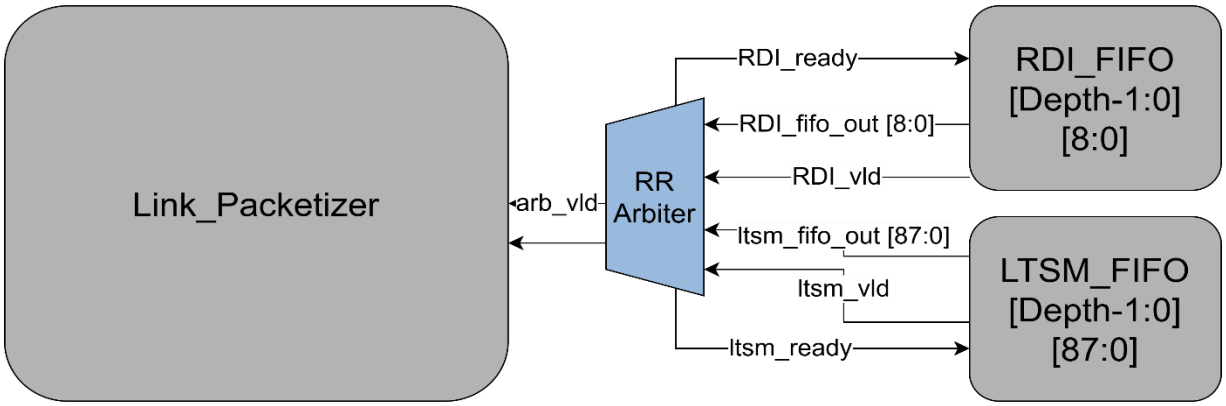


Figure 5-13: Round and Roben Arbiter Diagram

Table 5-10: RR Arbiter Interface signal

Signal	Direction	Width	Description
clk	Input	1	Sideband parallel clock (clk_sb)
rst_n	Input	1	Active-low reset
ltsm_valid	Input	1	LTSM async FIFO read-side valid
ltsm_data	Input	88	LTSM FIFO read data (msg_no[7:0], MsgInfo[15:0], data[63:0])
ltsm_ready	Output	1	LTSM FIFO pop strobe (dequeue enable)
rdi_valid	Input	1	RDI async FIFO read-side valid
rdi_data	Input	88	RDI FIFO read data (padded: {79'b0, stall, msg_no[7:0]})
rdi_ready	Output	1	RDI FIFO pop strobe
link_ready	Input	1	Downstream (Packetizer) ready
link_valid	Output	1	Arbiter output valid

Signal	Direction	Width	Description
link_data	Output	88	Selected message fields to Packetizer

5.4.4 Demux

The Training_Mgmt_Demux routes decoded packet fields from the DePacketizer to the correct receive-side interface. It inspects the decoded message number to classify the received packet as either an LTSM training message or an RDI link management message, then asserts the corresponding valid output. LTSM messages are presented on the ltsm_vld_rcvd path with their associated message number, data field, and MsgInfo field. RDI messages are presented on the RDI_vld_rcvd path with the decoded message number and stall flag. The demux is purely combinational; correct routing depends on the message number encoding defined in sb_pkg and msg_codec_pkg.

Table 5-11: Demux Interface signals

Signal	Direction	Width	Description
rx_msg_no_raw	Input	8	Decoded message number from DePacketizer
rx_msginfo_raw	Input	16	Decoded MsgInfo field
rx_payload_raw	Input	64	Decoded payload field
rx_vld_raw	Input	1	Decoded fields valid
rx_stall_raw	Input	1	Decoded stall flag
RDI_vld_rcvd	Output	1	RDI message received valid
RDI_msg_no_rcvd	Output	8	Received RDI message number
stall_rcvd	Output	1	Received RDI stall flag
ltsm_vld_rcvd	Output	1	LTSM message received valid
ltsm_msg_no_rcvd	Output	8	Received LTSM message number
msg_data_rcvd	Output	64	Received LTSM message data field
msg_info_rcvd	Output	16	Received LTSM MsgInfo field

5.4.5 Asynchronous FIFOs

Two asynchronous FIFOs within Training_Mgmt bridge the clock domain boundary between the lclk-domain sources in MainSM and the clk_sb-domain processing logic. The LTSM TX FIFO is 88 bits wide and 16 entries deep; it captures outbound LTSM messages (msg_no, MsgInfo, and data packed into 88 bits) written by the LTSM on the clk_ltsm write port and presents them to the RR_arbiter on the clk_sb read port. The RDI TX FIFO is 9 bits wide (msg_no plus stall bit) and 16 entries deep, with its write port in the clk_main domain and its read port in the clk_sb domain.

Both FIFOs use Gray-coded read and write pointers for safe CDC and provide WREADY and RVALID handshake signals to the respective domain logic. The receive-direction decoded outputs from the DePacketizer and Demux operate directly in the clk_sb domain and are consumed by MainSM through signals already synchronized at the MainSM boundary.

Table 5-12: Asynchronous FIFOs Interface singals

Signal	Direction	Width	Description
W_CLK	Input	1	Write-domain clock (clk_ltsm for LTSM FIFO; clk_main for RDI FIFO)
WRST_N	Input	1	Write-domain active-low reset
WINC	Input	1	Write enable (push strobe from MainSM)
WR_DATA	Input	88	Write data (88-bit LTSM word or 9-bit RDI word, zero-padded)
WFULL	Output	1	FIFO full indicator in write clock domain
WREADY	Output	1	FIFO ready to accept a new entry (not full)
R_CLK	Input	1	Read-domain clock (clk_sb)
RRST_N	Input	1	Read-domain active-low reset
RINC	Input	1	Read enable (pop strobe from RR_arbiter)
RD_DATA	Output	88	Read data to RR_arbiter
REMPTY	Output	1	FIFO empty indicator in read clock domain
RVALID	Output	1	Read data valid

5.5 RDI Controller

RDI_control is the sub-block responsible for transporting adapter configuration traffic between the D2D Adapter and the sideband link, and for routing register-access transactions to the Reg_Access block. It manages two independent data paths: a downstream path that carries adapter-originated packets from the 32-bit lp_cfg interface to the Link_Controller or Reg_Access, and an upstream path that carries PHY-originated or link-received packets back to the adapter through the 32-bit pl_cfg interface. Credit-based flow control governs the upstream request path to prevent the adapter's receive buffer from being overrun.

RDI_control also asserts the traffic_req signal to the RDI_SM within MainSM whenever upstream data is pending in the upstream request FIFO and no traffic window has been granted. This handshake coordinates upstream register-access responses and completions with the RDI state

machine's current operational state, ensuring that upstream traffic is not dispatched while the RDI_SM is in a state that does not permit it.

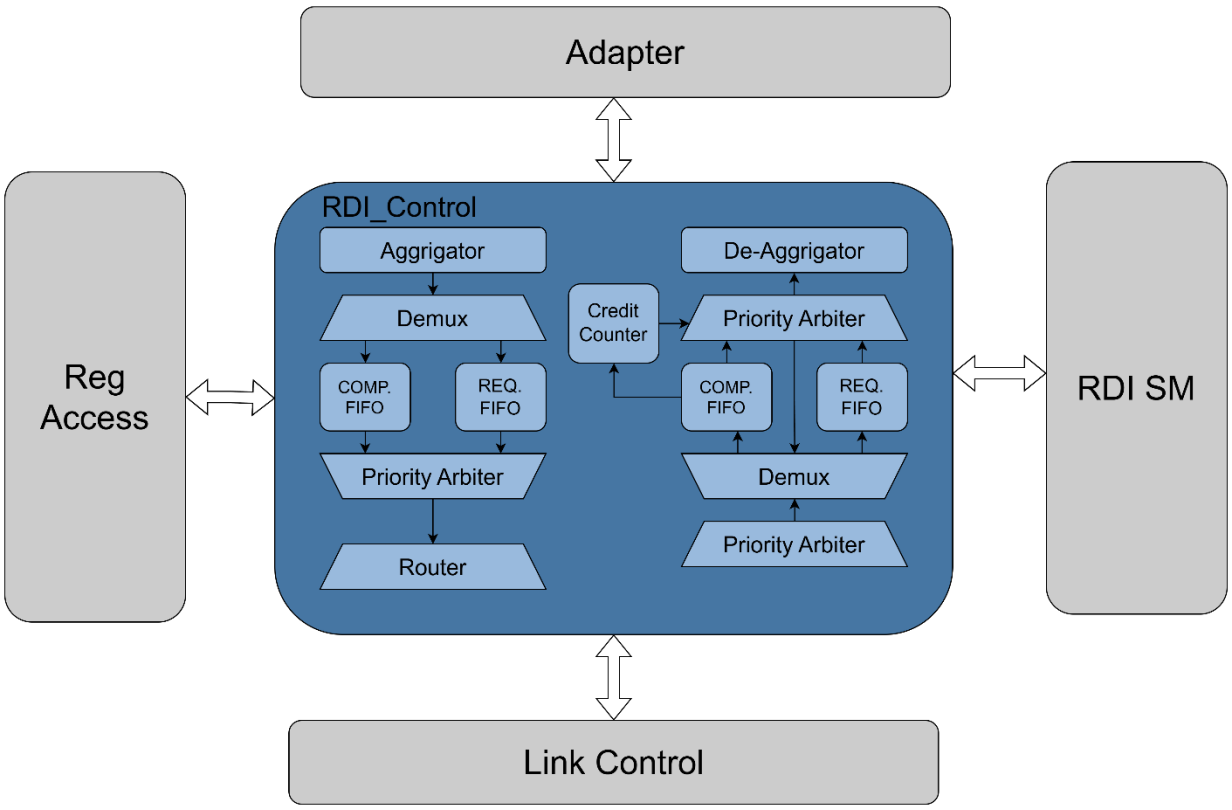


Figure 5-14: RDI Control diagram

5.5.1 Downstream Path

The downstream path processes adapter-originated traffic from the 32-bit lp_cfg interface to the 128-bit sideband packet domain. The rdi_aggregator reassembles consecutive 32-bit lp_cfg beats into a single 128-bit SB packet by accumulating beats until a complete packet boundary is reached. The assembled packet is classified by a rdi_comp_req_decoder instance, which inspects the opcode field to determine whether the packet is a completion or a request; completions carry higher priority in the subsequent arbitration. A sb_demux separates requests and completions into two independent synchronous FIFOs, from which a sb_priority_arbiter selects the next packet to forward. The rdi_router then determines the final destination: packets addressed to the local PHY register space — and any remote register requests received while phy_in_reset is asserted — are forwarded to Reg_Access through the reg_msg interface and the remote messages and remote completion is discarded while phy_in_reset is asserted; all other packets are forwarded to the Link_Controller through adapter_msg_send. One pl_cfg_crd pulse is generated for each request consumed from the downstream request FIFO, returning credit to the adapter.

5.5.2 Upstream Path

The upstream path carries packets destined for the adapter from two sources: packets received from the remote die through Link_Controller (adapter_msg_rcvd), and completion packets generated locally by Reg_Access (completion_msg). A priority arbiter merges these two sources, giving Link_Controller traffic higher priority to prevent blocking of link-received packets. The merged stream is classified by a second rdi_comp_req_decoder instance into completions and requests, which are separated into two upstream FIFOs. An upstream priority arbiter selects the next packet to forward, giving completions precedence and gating request forwarding on the availability of adapter credits tracked by a credit_counter. The credit_counter decrements on each upstream request forwarded and increments on each lp_cfg_crd pulse received from the adapter. The rdi_de_aggregator breaks each selected 128-bit packet into consecutive 32-bit pl_cfg beats for delivery to the adapter, asserting pl_cfg_vld on each valid beat.

5.6 Register Access

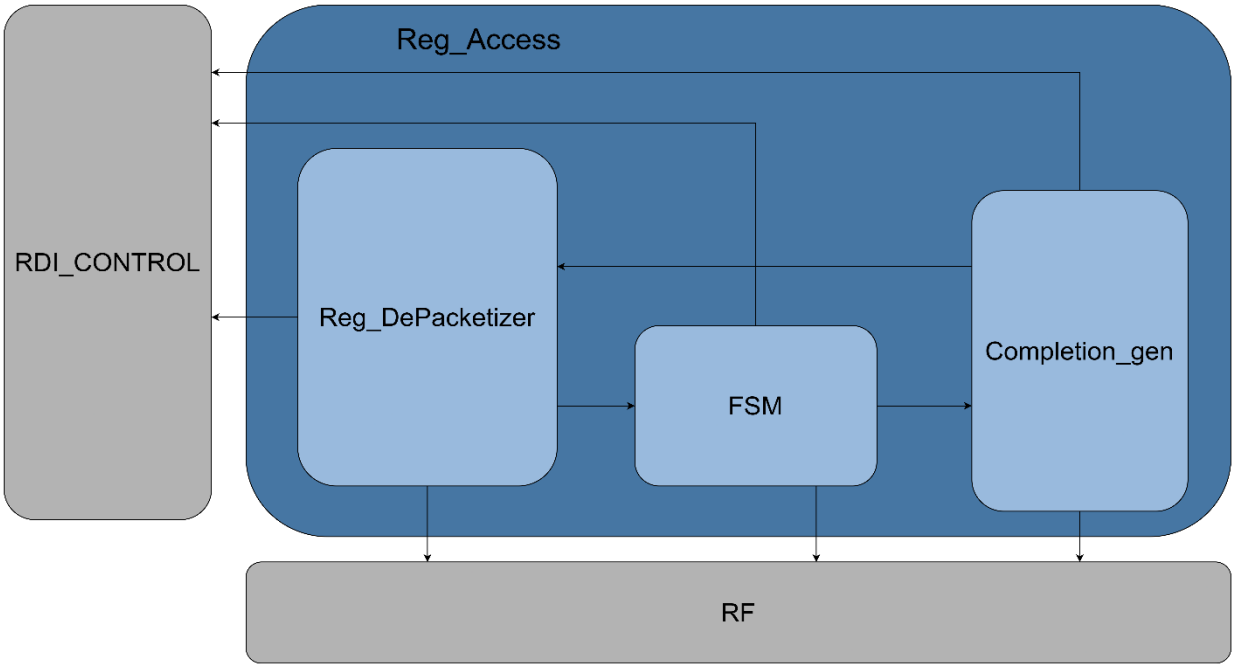


Figure 5-15: Reg Access block diagram

The Reg_Access block implements the register-access endpoint of the sideband subsystem. It receives inbound 128-bit register-access SB packets from RDI_control, extracts the register address, byte enables, access width, and write data, drives the Reg_File bus to execute the access, and generates a UCIe-compliant completion packet for return through RDI_control. The block is structured as three cooperating sub-modules — Reg_DePacketizer, Reg_Access_FSM, and Completion_gen — with a clear separation between the datapath and the control path. Reg_DePacketizer handles all packet parsing and field extraction; Reg_Access_FSM sequences the

access through its four states; and Completion_gen constructs the 128-bit completion packet after the access is resolved.

Table 5-13: Register Access Interface signals

Signal	Direction	Width	Description
clk	Input	1	Sideband parallel clock (clk_sb)
rst_n	Input	1	Active-low reset
reg_msg	Input	128	Incoming 128-bit register-access SB packet from RDI_control
reg_vld	Input	1	Packet valid
reg_rdy	Output	1	Block ready to accept a new packet
completion_msg	Output	128	Outgoing 128-bit completion SB packet to RDI_control
completion_vld	Output	1	Completion valid
completion_rdy	Input	1	RDI_control upstream arbiter ready to accept completion
rf_addr	Output	25	Register file address ({space[24], RL[23:20], offset[19:0]})
rf_be	Output	8	Byte enables for 64-bit access window
rf_is_64b_access	Output	1	64-bit access indicator
rf_wdata	Output	64	Write data for register write operations
rd_en	Output	1	Register file read enable
wr_en	Output	1	Register file write enable
rf_rdata	Input	64	Read data returned by Reg_File
rdata_vld	Input	1	Read data valid pulse from Reg_File
addr_err_o	Input	1	Address decode error from Reg_File; causes UR completion

5.6.1 Reg Depacketizer

The Reg_DePacketizer parses the incoming 128-bit SB packet and separates its content into control information for the FSM and datapath information for the Reg_File bus and Completion_gen. From the 64-bit header it extracts the opcode, source and destination identifiers, control parity bit, byte enables, access tag, data poison flag, and the 25-bit register address. It verifies the control parity field and asserts parity_err if the check fails. The ep (error poison) and false_msg flags are derived from the parsed header, allowing the FSM to determine the appropriate completion status without re-examining the raw packet. The decoded rf_addr, rf_be, rf_is_64b_access, and rf_wdata fields are driven directly onto the Reg_File interface wires. The original 64-bit header word is preserved as Original_Header for use by Completion_gen in constructing the completion, enabling correct tag and identifier mirroring in the response.

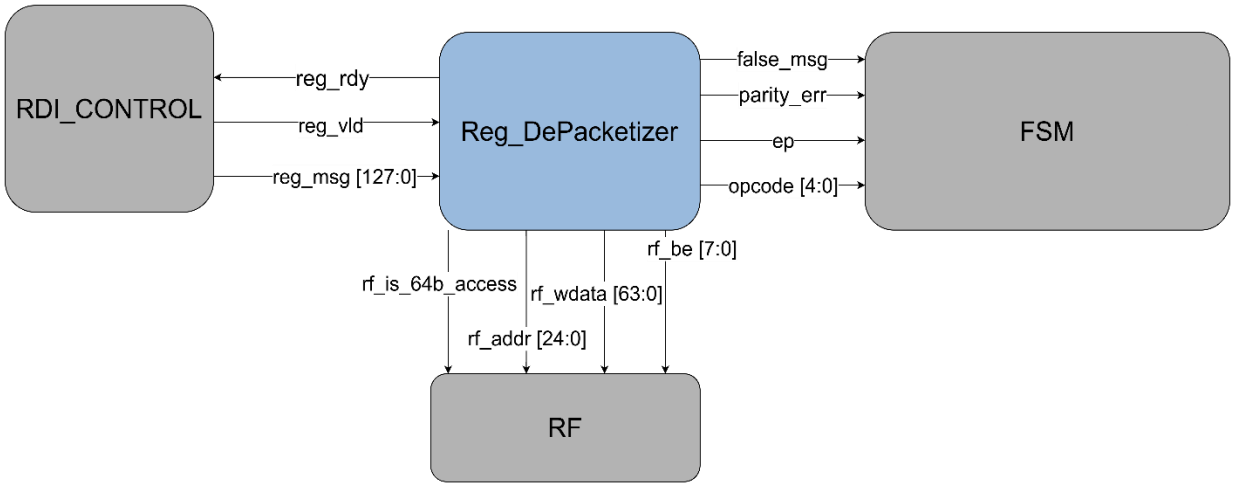


Figure 5-16: Reg Depacketizer Diagram

Table 5-14: Reg Depacketizer Interface signals

Signal	Direction	Width	Description
clk	Input	1	Sideband parallel clock (clk_sb)
rst_n	Input	1	Active-low reset
reg_msg	Input	128	128-bit incoming register-access SB packet
reg_vld	Input	1	Packet valid
reg_rdy	Output	1	Ready to accept packet (driven by FSM)
opcode	Output	5	Decoded 5-bit opcode from header
parity_err	Output	1	Control parity check failure indicator
ep	Output	1	Error poison bit from header
false_msg	Output	1	Unsupported or malformed message type indicator
rf_addr	Output	25	Register address decoded from packet header
rf_be	Output	8	Byte enables decoded from packet header
rf_is_64b_access	Output	1	64-bit access flag decoded from opcode
rf_wdata	Output	64	Write data decoded from packet payload field
Original_Header	Output	64	Preserved original 64-bit header word for completion generation

5.6.2 Reg Access FSM

The Reg_Access_FSM sequences the register-access operation through four states: IDLE, DECODE, EXECUTE, and GEN. In IDLE the FSM waits for reg_vld to be asserted. On receiving a valid packet, it transitions to DECODE and classifies the opcode as a read or write and checks all error flags. If

parity_err, ep, or false_msg is set, the FSM transitions directly to GEN with a status of Unsupported Request (UR). For a clean write, it asserts wr_en in EXECUTE for a single cycle then transitions to GEN. For a clean read, it asserts rd_en in EXECUTE and waits for rdata_vld to be returned by the register file before transitioning to GEN. In GEN it asserts completion_start to Completion_gen and holds until the upstream arbiter in RDI_control acknowledges with completion_rdy, then returns to IDLE. An addr_err_o assertion from the register file during EXECUTE overrides the success

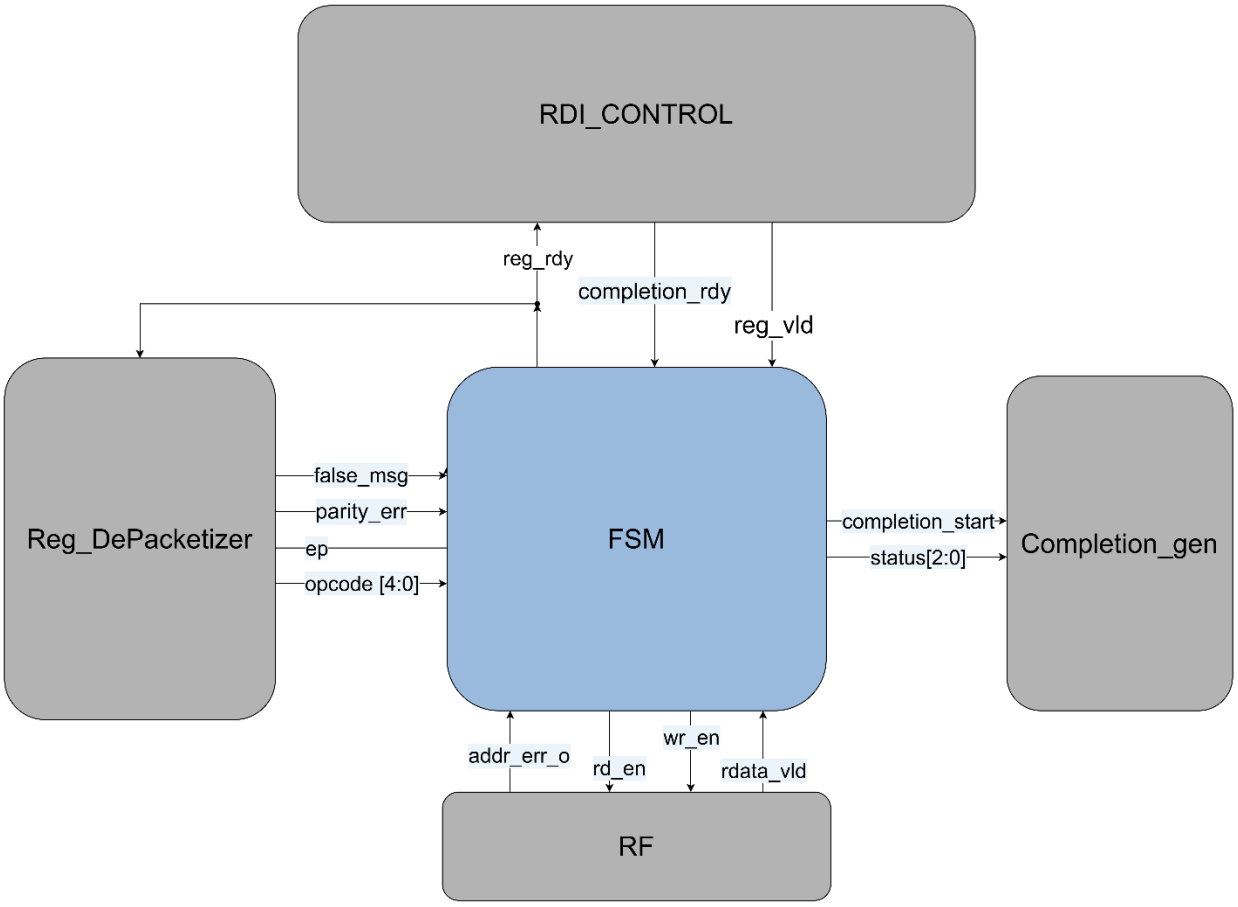


Figure 5-17: FSM Diagram

outcome and causes a UR completion.

Table 5-15: Reg Access FSM Interface signals

Signal	Direction	Width	Description
clk	Input	1	Sideband parallel clock (clk_sb)
rst_n	Input	1	Active-low reset
reg_vld	Input	1	Valid register-access packet present at DePacketizer input
reg_rdy	Output	1	FSM ready to accept a new request (IDLE state)
completion_rdy	Input	1	TX upstream arbiter ready to accept completion packet
opcode	Input	5	Decoded opcode from Reg_DePacketizer
parity_err	Input	1	Control parity error from Reg_DePacketizer
ep	Input	1	Error poison flag from Reg_DePacketizer
false_msg	Input	1	Invalid message type flag from Reg_DePacketizer
rd_en	Output	1	Register file read enable (asserted in EXECUTE for reads)
wr_en	Output	1	Register file write enable (asserted in EXECUTE for writes)
rdata_vld	Input	1	Register file read data valid (exits EXECUTE wait)
rf_addr_err	Input	1	Address decode error from register file
status	Output	3	Completion status code (3'b000 = SC, 3'b001 = UR)
completion_start	Output	1	Triggers Completion_gen to begin packet construction

5.6.3 Register File Bus Interface

The register file bus interface is the set of signals exchanged between Reg_Access and the Reg_File subsystem. The 25-bit rf_addr encodes the target register using the UCIe-defined address format: bit 24 selects the Config Space window (0) or the MMIO PHY block window (1), bits 23:20 carry the Register Locator (RL) field, and bits 19:0 carry the byte offset within the selected space. The rf_be field carries up to eight byte-enable bits covering the full 64-bit access window. The rf_is_64b_access flag allows the register file to independently determine the access granularity from the byte-enable pattern. Write data is driven on rf_wdata for write operations. Read data is returned on rf_rdata with a rdata_vld pulse once the access is complete. The addr_err_o signal is asserted by the register file when the presented address does not map to any implemented register, causing the FSM to generate an Unsupported Request completion in response.

Table 5-16: Register File Bus Interface signals

Signal	Direction	Width	Description
rf_addr	Output	25	Register address ({space[24], RL[23:20], offset[19:0]})
rf_be	Output	8	Byte enables for the 64-bit access word
rf_is_64b_access	Output	1	64-bit access indicator (derived from opcode)
rf_wdata	Output	64	Write data for register write operations
rd_en	Output	1	Read enable; initiates a register read in Reg_File
wr_en	Output	1	Write enable; initiates a register write in Reg_File
rf_rdata	Input	64	Read data word returned by Reg_File
rdata_vld	Input	1	Single-cycle valid pulse when rf_rdata is ready
addr_err_o	Input	1	Address decode error; unmapped address on rf_addr

5.6.4 Completion Generator

The Completion_gen sub-block constructs the 128-bit UCIe completion packet returned to the requesting adapter or remote die following a register access. It receives the Original_Header word preserved by Reg_DePacketizer and mirrors the access tag, source-destination identifier swap, and other header fields required by the UCIe completion format. The completion status field is populated from the status code driven by Reg_Access_FSM: Successful Completion (SC, 3'b000) for a clean access, or Unsupported Request (UR, 3'b001) for an address error, parity error, error poison, or unsupported opcode. For read completions, the rf_rdata value latched when rdata_vld is asserted is embedded in the 64-bit payload field of the completion packet. For write completions and UR completions, the payload field is set to zero. The constructed 128-bit completion packet is presented on completion_msg with completion_vld asserted and held until the upstream arbiter in RDI_control acknowledges acceptance through completion_rdy.

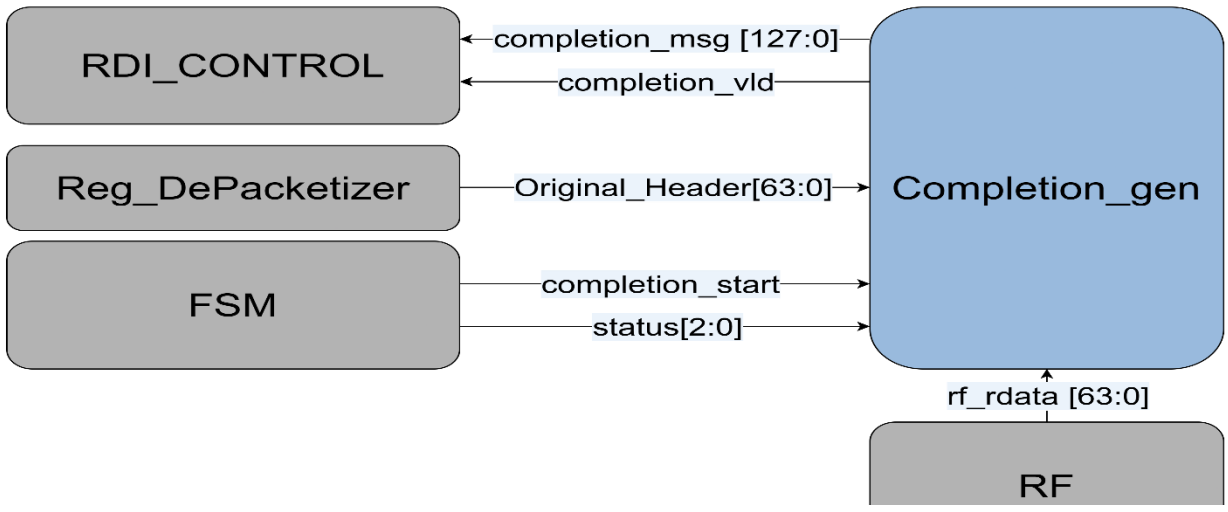


Figure 5-18: Completion Generator Diagram

Table 5-17: Completion Generator Interface signals

Signal	Direction	Width	Description
clk	Input	1	Sideband parallel clock (clk_sb)
rst_n	Input	1	Active-low reset
completion_start	Input	1	Trigger from Reg_Access_FSM to begin packet construction
status	Input	3	Completion status (3'b000 = SC, 3'b001 = UR)
Original_Header	Input	64	Original 64-bit request header (for tag and ID mirroring)
rf_rdata	Input	64	Register file read data embedded in read-completion payload
rdata_vld	Input	1	Latches rf_rdata into the completion payload
completion_msg	Output	128	Constructed 128-bit completion SB packet
completion_vld	Output	1	Completion packet valid
completion_rdy	Input	1	Upstream arbiter ready to accept completion

6.0 MainSM Design

6.1 MainSM Wrapper Overview

MainSM is the top-level integration wrapper that groups the two link-state machines of the UCIe PHY into a single subsystem boundary. It instantiates LTSM_TOP (u_ltsm_top) and RDI_SM (u_rdi_sm) as peer sub-blocks and adds a minimal amount of inter-machine glue that logically belongs between them. The two sub-blocks share no clock domain crossing; both operate in the lclk domain, with LTSM_TOP receiving the gated variant (gated_lclk) while RDI_SM and the glue flops run on the ungated lclk.

The glue logic retained inside MainSM covers three concerns. First, mapper-enable gating: the LTSM produces ltsm_mapper_en to enable the Adapter data path, but during an RDI stall the data path must be held off even when the LTSM considers itself in ACTIVE. The stall_done_latched_w level, produced by the RDI_SM stall handshake unit and held until mapper_en falls, is ANDed with ltsm_mapper_en to generate the final mb_mapper_en signal driven to unit_mb_die. Second, a sticky SBINIT pattern-detected flop captures the det_pat_rcvd pulse from SideBand_Top and holds it as sticky_sb_pattern_detected across the boundary to the RDI_SM gating logic, which needs this level rather than the pulse. Third, the phy_rm_link_err_i decode: when the RDI_SM receives a link-management message whose encoding corresponds to a remote link error, MainSM decodes this and exposes it as phy_rm_link_err_i for the error log registers. Additionally, MainSM performs msg_no_e-to-bus and bus-to-msg_no_e casts on the sideband message identifiers before forwarding them to the LTSM and RDI_SM respectively, since the two sub-blocks use enumerated types while the external SideBand_Top interface uses raw logic vectors.

6.2 LTSM

The Link Training State Machine is implemented in LTSM_wrapper.sv, which is in turn wrapped by LTSM_TOP.sv together with the mainband_ltsm_interface signal translation layer. LTSM_wrapper integrates the central sequencing controller (unit_ltsm_controller), a shared 8 ms watchdog timer, all top-level state sub-blocks, the trainerror_handshake module, state log registers, and the SB/MB output multiplexers that route the active state's sideband and mainband control outputs outward. The entire LTSM operates in the gated_lclk domain.

6.2.1 LTSM Architecture

The architecture of the LTSM is built around a strict separation between sequencing and behavior. The unit_ltsm_controller is the only component that determines which state is active at any moment; it holds no datapath logic and makes transitions solely on the done, error, and timeout pulses presented by each state sub-block. Each state sub-block is enabled by a single enable pulse from the controller and reports completion or failure through a corresponding done/error output. The controller activates exactly one state block at a time. The LTSM_wrapper multiplexes the SB

TX and mainband control outputs of all state blocks behind the current enable, so only the active block drives the shared buses.

The 8 ms watchdog (timeout_counter) is enabled whenever the LTSM is in SBINIT, MBINIT, MBTRAIN, LINKINIT, or PHYRETRAIN. The counter is restarted on every substate transition, so each individual sub-state receives its own fresh 8 ms budget rather than sharing a single budget across the entire parent state. A timeout in SBINIT or a second consecutive timeout in any active training state drives an immediate transition to CTRL_TRAINERROR. A timeout in MBINIT, MBTRAIN, or LINKINIT arms the TRAINERROR-entry handshake (te_local_error_trigger), and the controller advances to CTRL_TRAINERROR only after te_handshake_done is received from the trainerror_handshake module.

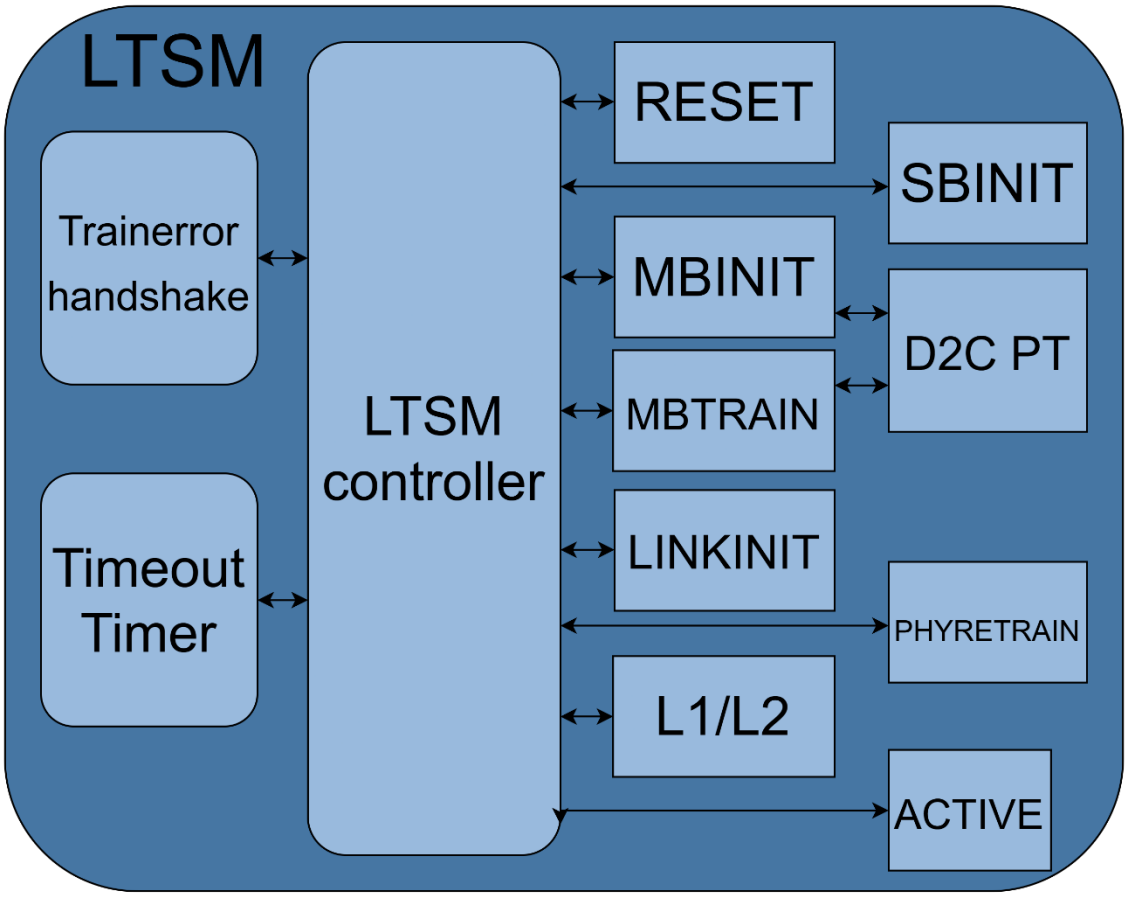


Figure 6-1: LTSM Block Diagram

The LTSM maintains a 4-entry shift-register state log. On every substate transition where the state-log code changes, the log shifts: the new code enters log0_state_n, and the three preceding codes shift into log0_state_n_minus_1, log0_state_n_minus_2, and log1_state_n_minus_3. The log also captures the lane reversal and width degradation flags at the moment of the state-N entry, providing the register file with a four-state history that persists until the next state change.

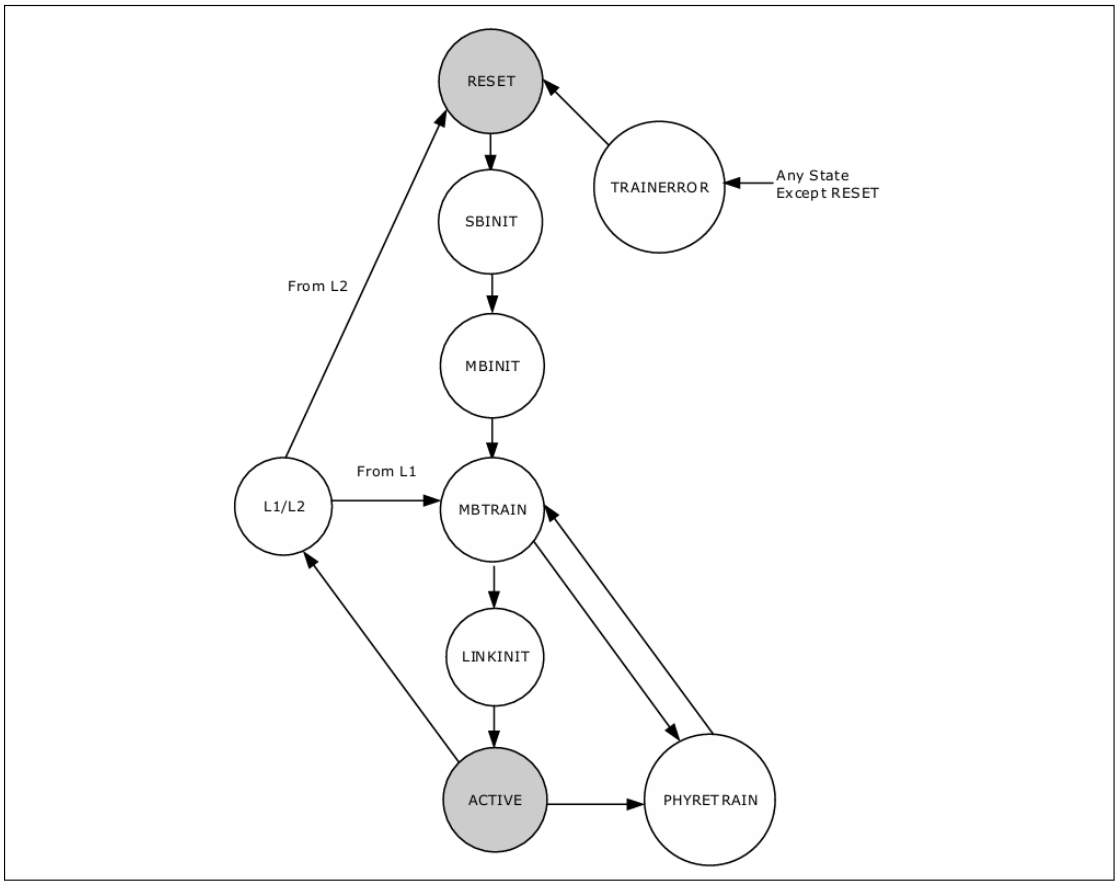


Figure 6-2: LTSM State diagram

6.2.2 LTSM Controller

unit_ltsm_controller is a pure FSM with no datapath. It sequences all ten LTSM states through a registered next-state logic block that evaluates one set of transitions per state. The nominal linear path is RESET → SBINIT → MBINIT → MBTRAIN → LINKINIT → ACTIVE. Each step requires the current state's done signal; each step gates any error from that state into a direct or handshake-mediated transition to CTRL_TRAINERROR.

Several non-linear transitions are also encoded. From CTRL_ACTIVE, the next state is resolved by the active_next_ltsm_state input, which the ACTIVE sub-block drives to CTRL_PHYRETRAIN, CTRL_L1, CTRL_L2, or CTRL_TRAINERROR depending on the condition observed. CTRL_PHYRETRAIN exits to CTRL_MBTRAIN on phyretrain_done. CTRL_L1 exits to CTRL_MBTRAIN

with mbtrain_speedidle_req latched, so MBTRAIN re-enters at the SPEEDIDLE substate, bypassing the voltage reference calibration substates that were already completed during initial bring-up. CTRL_L2 exits to CTRL_RESET on l2_done, requiring a full retraining sequence. CTRL_TRAINERROR always exits to CTRL_RESET on trainerror_done. If the RDI_SM enters LinkError, LinkReset, or Disabled at any time, the controller transitions directly to CTRL_TRAINERROR, treating any of these as a fatal link condition.

Table 6-1: LTSM Controller Interface Signals

Signal	Direction	Width	Description
clk	Input	1	Sideband-gated lclk (gated_lclk); clocks the FSM register
rst_n	Input	1	Active-low reset
reset_en	Output	1	Enable pulse to the RESET state sub-block
reset_done	Input	1	RESET state complete (4 ms dwell + training-start satisfied)
sbinit_en	Output	1	Enable pulse to the SBINIT state sub-block
sbinit_done	Input	1	SBINIT complete (SB channel established)
sbinit_error	Input	1	SBINIT error — direct jump to CTRL_TRAINERROR (no handshake)
mbinit_en	Output	1	Enable pulse to MBINIT_WRAPPER
mbinit_done	Input	1	MBINIT complete (all six substates finished)
mbinit_error	Input	1	MBINIT error — arms the §4.5.3.8 TRAINERROR-entry handshake
mbtrain_en	Output	1	Enable pulse to wrapper_MBTRAIN
mbtrain_done	Input	1	MBTRAIN complete → next state is LINKINIT
mbtrain_error	Input	1	MBTRAIN error — arms the TRAINERROR-entry handshake
mbtrain_phyretrain_req	Input	1	MBTRAIN requests PHYRETRAIN (from wrapper_MBTRAIN LINKSPEED exit)
mbtrain_speedidle_req	Output	1	Latched flag to wrapper_MBTRAIN: re-enter at SPEEDIDLE (L1 exit)
linkinit_en	Output	1	Enable pulse to the LINKINIT state sub-block
linkinit_done	Input	1	LINKINIT complete → transition to CTRL_ACTIVE
linkinit_error	Input	1	LINKINIT error — arms the TRAINERROR-entry handshake
active_en	Output	1	Enable to the ACTIVE state sub-block (level, not pulse)

Signal	Direction	Width	Description
active_next_ltsm_state	Input	4	ACTIVE-resolved exit state (PHYRETRAIN / L1 / L2 / TRAINERROR)
active_error	Input	1	ACTIVE detected a fatal error (from pl_error path)
phyretrain_en	Output	1	Enable pulse to the PHYRETRAIN state sub-block
phyretrain_done	Input	1	PHYRETRAIN complete → transition to CTRL_MBTRAIN
l1_en	Output	1	Enable pulse to the L1 state sub-block
l1_done	Input	1	L1 exit → transition to CTRL_MBTRAIN (SPEEDIDLE re-entry)
l1_error	Input	1	L1 error → CTRL_TRAINERROR
l2_en	Output	1	Enable pulse to the L2 state sub-block
l2_done	Input	1	L2 exit → transition to CTRL_RESET
l2_error	Input	1	L2 error → CTRL_TRAINERROR
trainerror_en	Output	1	Enable pulse to the TRAINERROR state sub-block
trainerror_done	Input	1	TRAINERROR complete → return to CTRL_RESET
te_handshake_done	Input	1	§4.5.3.8 two-way SB TRAINERROR handshake complete; gates entry to CTRL_TRAINERROR from MBINIT/MBTRAIN/LINKINIT
timeout_8ms_occured	Input	1	8 ms watchdog expired in current training state
second_timeout_occured	Input	1	Second consecutive 8 ms timeout → immediate CTRL_TRAINERROR
current_ltsm_state	Output	Enum	Current LTSM top-level state (LTSM_state_e); exported for SB phy_in_reset and observability
rdi_state	Input	Enum	Current RDI SM state; LinkError/LinkReset/Disabled triggers CTRL_TRAINERROR from any LTSM state

6.2.3 RESET

The RESET sub-block enforces a minimum 4 ms dwell period after every entry to RESET, allowing PLLs and analog front-end circuits to stabilize. The 4 ms timer is internal to the RESET block. The block does not assert reset_done until both the 4 ms period has elapsed and phy_start_ucie_link_training_ctrl_out has been asserted by software through the Reg_File control register. During RESET the mainband transmitters are tri-stated, the sideband subsystem is held in reset mode (phy_in_reset asserted), and no training activity is performed. RESET may be entered from TRAINERROR or from L2 exit or from primary reset deassertion. After reset_done, the controller transitions to SBINIT.

Table 6-2: RESET interface signals

Signal	Direction	Width	Description
lclk	Input	1	RDI domain clock.
rst_n	Input	1	Active-low reset.
mb_tx_clk_lane_sel	Output	2	Tri-state code (1xb); Clock transmitter tri-stated.
mb_tx_data_lane_sel	Output	2	Tri-state code (1xb); Data transmitters tri-stated.
mb_tx_val_lane_sel	Output	2	Tri-state code (1xb); Valid transmitter tri-stated.
mb_tx_trk_lane_sel	Output	2	Tri-state code (1xb); Track transmitter tri-stated.
mb_rx_clk_lane_sel	Output	1	Receiver permitted to be disabled (0b).
mb_rx_data_lane_sel	Output	1	Receiver permitted to be disabled (0b).
mb_rx_val_lane_sel	Output	1	Receiver permitted to be disabled (0b).
mb_rx_trk_lane_sel	Output	1	Receiver permitted to be disabled (0b).
sb_tx_hold_low	Output	1	Holds the sideband transmitter low unless SB_MGMT_UP is set.
SB_MGMT_UP	Input	1	Indicates Management Transport path already established; relaxes the RESET-exit precondition.
pl_state_sts	Output	4	Held at Reset (0000b) for the duration of the state.
lp_state_req	Input	4	Monitored for a link-training trigger as part of the RESET-exit precondition.
clk_sb	Input	1	Sideband clock; must be running at 800 MHz as an exit precondition.
mb_pll_speed_sel	Output	4	Held at the lowest supported I/O data rate before RESET may be exited.
reg_training_start	Input	1	Firmware/software link-training trigger checked as an exit precondition.

6.2.4 SBINIT

The SBINIT sub-block orchestrates sideband initialization. It drives pattern_mode, start_pat_req, and sbinit_req_iter_count to SideBand_Top to initiate the 64-UI alternating clock pattern sequence, and it monitors iter_done and the sticky sb_det_pattern_rcvd (via sticky_sb_pattern_detected) to determine when the remote die has detected the pattern and the additional four iterations have been sent. Once pattern detection and iteration completion are both confirmed, the block transitions to the sideband message exchange phase, during which the {SBINIT Out of Reset}, {SBINIT done req}, and {SBINIT done resp} sideband messages are exchanged with the remote die to establish the sideband control channel. On successful completion of this

exchange, sbinit_done is asserted. Any timeout or SB message failure asserts sbinit_error, which the controller forwards directly to CTRL_TRAINERROR without a two-way handshake, since the sideband channel has not yet been established.

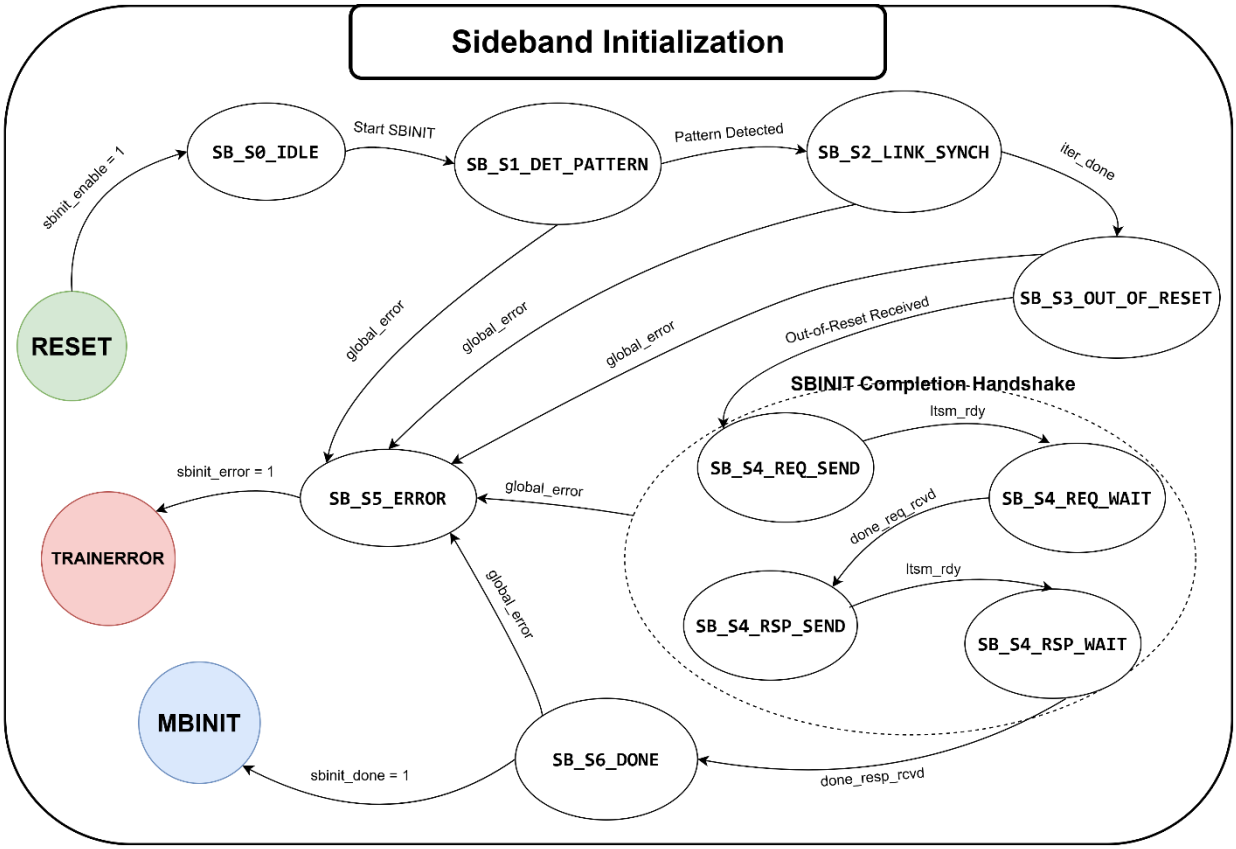


Figure 6-3: SBINIT state diagram

Table 6-3: SBINIT Interface signals

Signal	Direction	Width	Description
lclk / rst_n	Input	1 each	Clock and active-low reset.
sb_tx_clk / sb_tx_data	Output	1 each	Sideband serial transmit clock/data pins.
sb_rx_clk / sb_rx_data	Input	1 each	Sideband serial receive clock/data pins.
sbinit_pattern_ctrl	Output	Enum	Governs the sideband initialization pattern generation/detection sequence.
sbinit_done_req	Output	Enum	{SBINIT done req} message; announces local readiness to exit SBINIT.
sbinit_done_resp	Input	Enum	{SBINIT done resp} message; partner acknowledgment completing SBINIT.
SB_MGMT_UP	Input	1	Enables the Management-Transport SBINIT re-entry rules.
pl_state_sts	Output	4	Remains at Reset (0000b) throughout SBINIT.
sb_msg_timeout	Input	1	8 ms watchdog timeout; drives direct transition to TRAINERROR.

6.2.5 MBINIT

MBINIT is implemented as MBINIT_WRAPPER, which integrates MBINIT_CONTROLLER with six sub-state modules. The controller sequences the fixed substate chain PARAM → CAL → REPAIRCLK → REPAIRVAL → REVERSALMB → REPAIRMB. Each sub-state asserts its done or error output to the controller, which activates the next sub-state in sequence. The controller also owns the mainband output mux for the MBINIT phase: it forwards the TX pattern enable, pattern setup, val pattern select, RX compare enable, and RX compare setup signals from whichever substate is currently active into the shared wires that drive the LTSM wrapper's mainband output bus. Lane mask outputs (mbinit_rx_data_lane_mask, mbinit_tx_data_lane_mask) are accumulated and latched by the controller at the REPAIRMB stage, where the final negotiated lane map is determined.

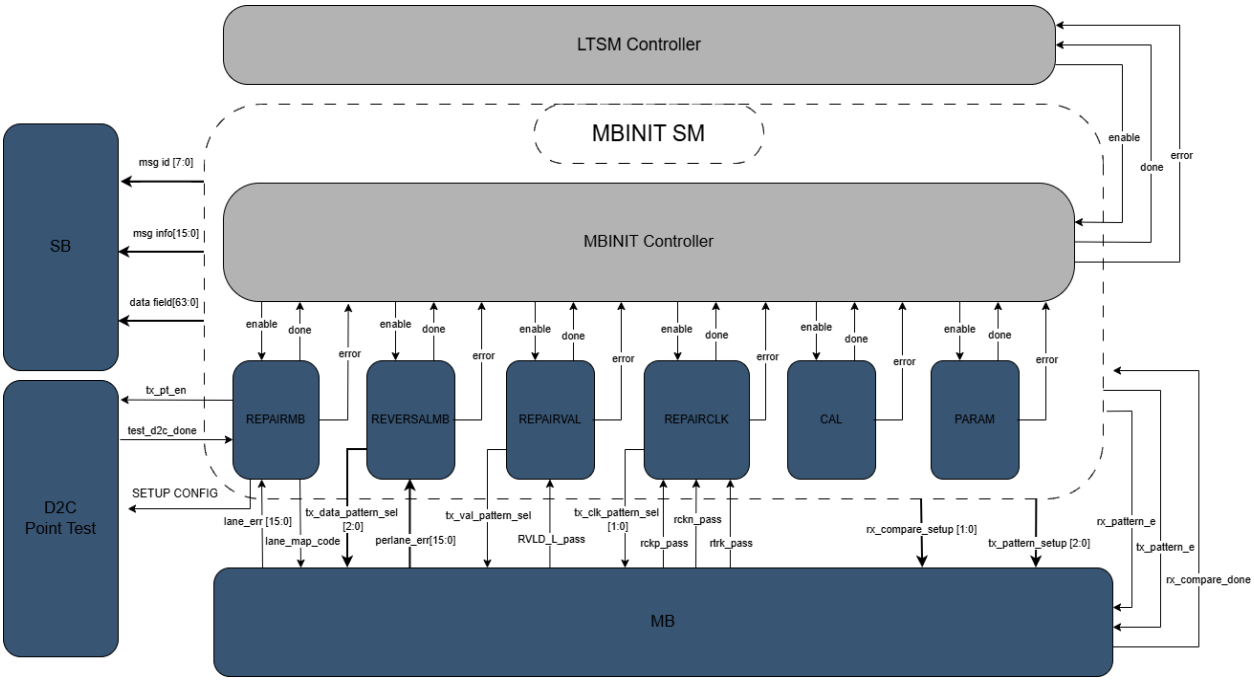


Figure 6-4: MBINIT Architecture

Table 6-4: MBINIT Interface signals

Signal	Direction	Width	Description
mbinit_en	Input	1	Enable from top-level LTSM controller.
mbinit_done / mbinit_error	Output	1 each	Completion and error handshake to the LTSM controller.
mbinit_state_n	Output	Enum	Current MBINIT substate, logged to Reg_File Error Log.
mb_lfsr_state	Output	2	LFSR mode select: clear, PRBS, per-lane ID, data.
mb_width_deg_tx/ mb_width_deg_rx	Output	3 each	Width-degradation lane-map code applied by REPAIRMB.
mb_reversal_en	Output	1	Lane reversal enable, set by REVERSALMB.
mb_mapper_en/ mb_demapper_en	Output	1 each	Byte-to-lane mapper/demapper enables.
mb_rx_data_deser_en/ mb_rx_valid_deser_en	Output	1 each	Receive deserializer enables.
mb_clk_pattern_en/ mb_clk_embedded_en/ mb_clk_detector_en	Output	1 each	Clock training-pattern, embedded-clock, and detector enables.

Signal	Direction	Width	Description
mb_valid_pattern_en	Output	1	Valid training-pattern generation enable.
mb_pcmp_enable / mode / mask / iter_cnt / clear	Output	1/2/16/8/1	Pattern-comparator configuration bus used by REPAIRMB.
mb_vcmp_enable/ mb_vcmp_mode/ mb_vcmp_clear	Output	1/1/1	Valid-comparator configuration used by REPAIRVAL.
mb_pll_speed_sel	Output	4	Held at 4 GT/s for the duration of MBINIT.
mb_lfsr_tx_done/ mb_valid_done/ mb_clk_done	Input	1 each	Pattern-completion indicators consumed for substate progression.
mb_pcmp_done/ mb_pcmp_per_lane_pass/ mb_pcmp_agg_error	Input	1/16/1	Pattern-comparator completion, per-lane pass vector, aggregate error.
mb_vcmp_done/ mb_vcmp_pass	Input	1/1	Valid-comparator completion and pass indication.
mb_clk_p_pass/ mb_clk_n_pass/ mb_track_pass	Input	1 each	Forwarded-clock and track detection results, consumed by REPAIRCLK.
mb_valid_frame_error	Input	1	Forwarded outward as pl_error during subsequent Active operation.
reg_Target_Link_Width_ctrl / reg_Target_Link_Speed_ctrl / reg_lane_mask	Input	4/4/16	Software configuration consumed by MBINIT.PARAM.

MBINIT.PARAM

The PARAM sub-state performs the mandatory parameter exchange with the remote die over the sideband channel. It constructs and transmits the {MBINIT.PARAM configuration req} message carrying the local transmitter voltage swing, maximum supported data rate, clock mode preference, Standard Package Module Width capability bit, SFES capability bit, and PMO support bit. It then receives the partner's {MBINIT.PARAM configuration resp} and resolves the negotiated operating parameters: the common maximum data rate (minimum of local and partner), the agreed clock mode, and the feature flags. If SFES is mutually supported, the sub-state proceeds with the two-stage SFES negotiation ({MBINIT.PARAM SBFE req}/{MBINIT.PARAM SBFE resp}), after which PMO and other feature bits are settled. The resolved parameters are held in internal registers and forwarded to the LTSM wrapper for distribution to the register file and other subsystems.

Table 6-5: MBINIT.PARAM Interface signals

Signal	Direction	Width	Description
param_config_req	Output	Enum	{MBINIT.PARAM configuration req}: Tx voltage swing, max data rate, clock-mode preference, SPMW cap bit, SFES cap bit.
param_config_resp	Input	Enum	{MBINIT.PARAM configuration resp}: reflects shared capabilities; common data rate = min(local, remote).
reg_Supported_TX_Vswing	Input	5	Local advertised transmitter voltage swing.
reg_Max_Link_Speed_cap	Input	4	Local advertised maximum data rate capability.
reg_Clock_mode_cap / reg_Clock_Phase_cap	Input	2/2	Clock mode/phase capability bits.
reg_Clock_mode_ctrl / reg_Clock_Phase_ctrl	Input	1/1	Clock mode/phase control bits.
reg_phy_x8_mode_ctrl / SPMW (strap)	Input	1/1	Standard Package Module Width capability (x8/x16).
reg_PMO_support_local_cap / ctrl	Input	1/1	Sideband Performant Mode capability/control bits.
reg_TARR_support_local_cap / ctrl	Input	1/1	TARR feature capability/control bits.
reg_L2SPD_support_local_cap / ctrl	Input	1/1	L2 Sideband Power Down capability/control bits negotiated per protocol; mechanism itself excluded from this design.
reg_PSPT_support_local_cap / ctrl	Input	1/1	Priority Sideband Packet Transfer capability/control bits; excluded feature, negotiated but not implemented.
mb_tx_*_lane_sel	Output	2 each	Mainband transmitters remain tri-stated for the duration of PARAM.
mb_rx_*_lane_sel	Output	1 each	Mainband receivers may be disabled for the duration of PARAM.

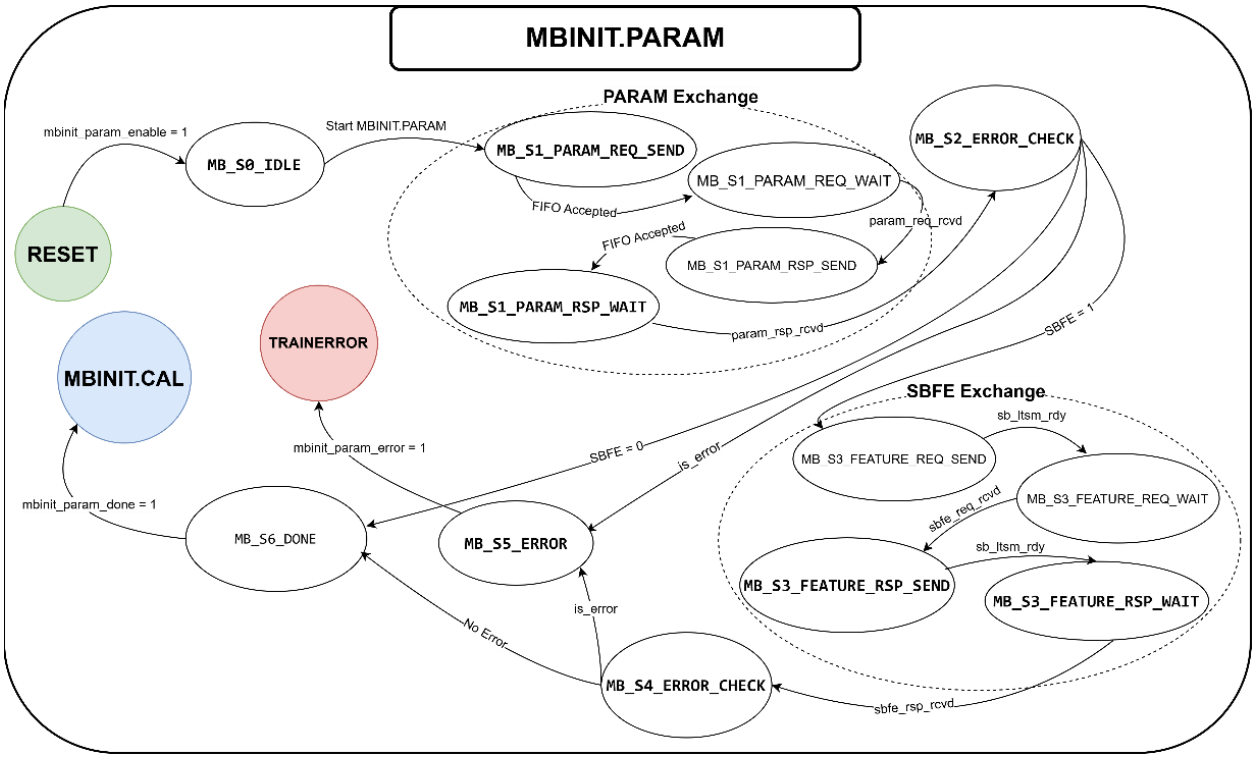


Figure 6-5: MBINIT.PARAM state diagram

MBINIT.CAL

The CAL sub-state performs on-die calibration at the lowest supported data rate of 4 GT/s. In the Standard Package, this consists of clock detection confirmation: the sub-state enables the clock pattern detector and verifies that the forwarded clock and track lanes are present and toggling at the expected rate. The calibration result is fed back from the mainband comparator logic as the clk_p_pass, clk_n_pass, and track_pass signals. On successful detection, cal_done is asserted. A failure asserts cal_error, triggering the TRAINERROR entry handshake.

Table 6-6: MBINIT.CAL Interface Signals

Signal	Direction	Width	Description
cal_en	Output	1	Enable for the on-die calibration interval.
cal_done	Input	1	Completion of on-die calibration; no sideband message is defined for this substate.
mb_pll_speed_sel	Output	4	Held at the lowest supported data rate during calibration.
mb_tx_*_lane_sel / mb_rx_*_lane_sel	Output	2/1 each	Transmitters remain tri-stated; receivers remain in the disabled posture inherited from PARAM.
phy_cal_en	Output	1	Digital control interface enable to the abstracted analog front-end calibration block.
phy_cal_done	Input	1	Analog front-end calibration-complete indication.

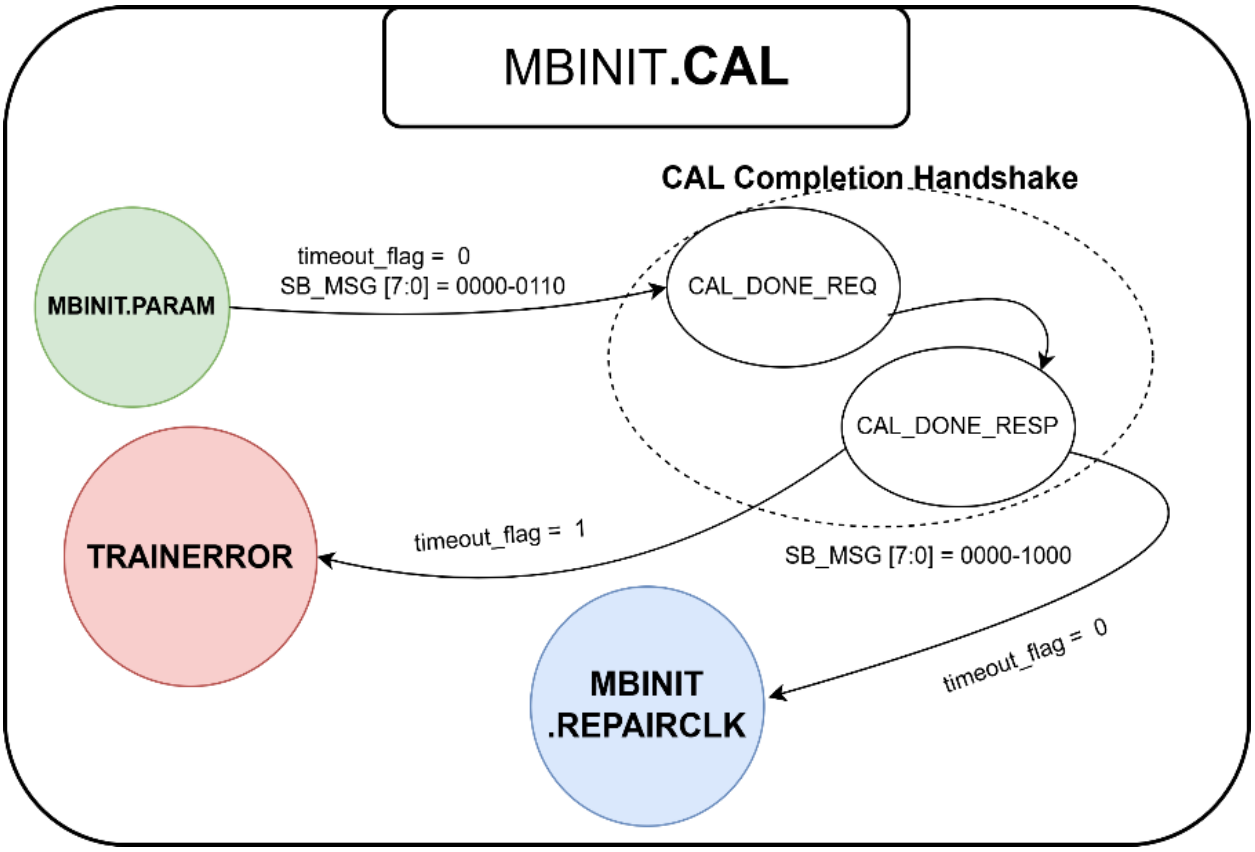


Figure 6-6: MBINIT.CAL state diagram

MBINIT.REPAIRCLK

The REPAIRCLK sub-state validates the integrity of the forwarded clock and track lanes at the lowest data rate. In the Standard Package, where no physical lane repair is possible, the sub-state drives the clock pattern transmitter and enables the clock pattern detector on the receive side, then coordinates with the remote die through sideband messages to confirm mutual clock and track reception. The repairclk_tx_pattern_en, repairclk_tx_pattern_setup, repairclk_rx_compare_en, and repairclk_rx_compare_setup signals are driven to the MBINIT controller's mainband output mux. Pass or fail results are captured from mb_clk_p_pass, mb_clk_n_pass, and mb_track_pass and reported to the controller via repairclk_done or repairclk_error.

Table 6-7: MBINIT.REPAIRCLK Interface Signals

Signal	Direction	Width	Description
repairclk_init_req	Output	Enum	{init req}: readiness handshake before clock repair pattern transmission.
repairclk_init_resp	Input	Enum	{init resp}: partner ready to receive pattern.
repairclk_result_req	Output	Enum	{result req}: requests logged per-lane compare results.
repairclk_result_resp	Input	Enum	{result resp}: RCKP_L/RCKN_L/RTRK_L/RRDCK_L compare bits.
repairclk_apply_repair_req	Output	4	{apply repair req}: repair encoding for the affected clock/track lane (Advanced Package only).
repairclk_apply_repair_resp	Input	Enum	{apply repair resp}: acknowledgment.
repairclk_check_repair_req / resp	Input/Output	Enum	{check repair init/results req/resp}: re-validates the repaired lane.
repairclk_done_req	Output	Enum	{done req}: completion handshake.
repairclk_done_resp	Input	Enum	{done resp}: closes the substate; transitions to REPAIRVAL.
mb_tx_clk_lane_sel / mb_tx_trk_lane_sel	Output	2/2	Drives 128 iterations of the unscrambled clock repair pattern.
mb_rx_clk_lane_sel / mb_rx_trk_lane_sel	Output	1/1	Enables detection of the received clock repair pattern (≥16 consecutive cycles required).
mb_clk_p_pass / mb_clk_n_pass / mb_track_pass	Input	1 each	Logged detection outcome; failure on any lane is non-recoverable in the Standard Package.

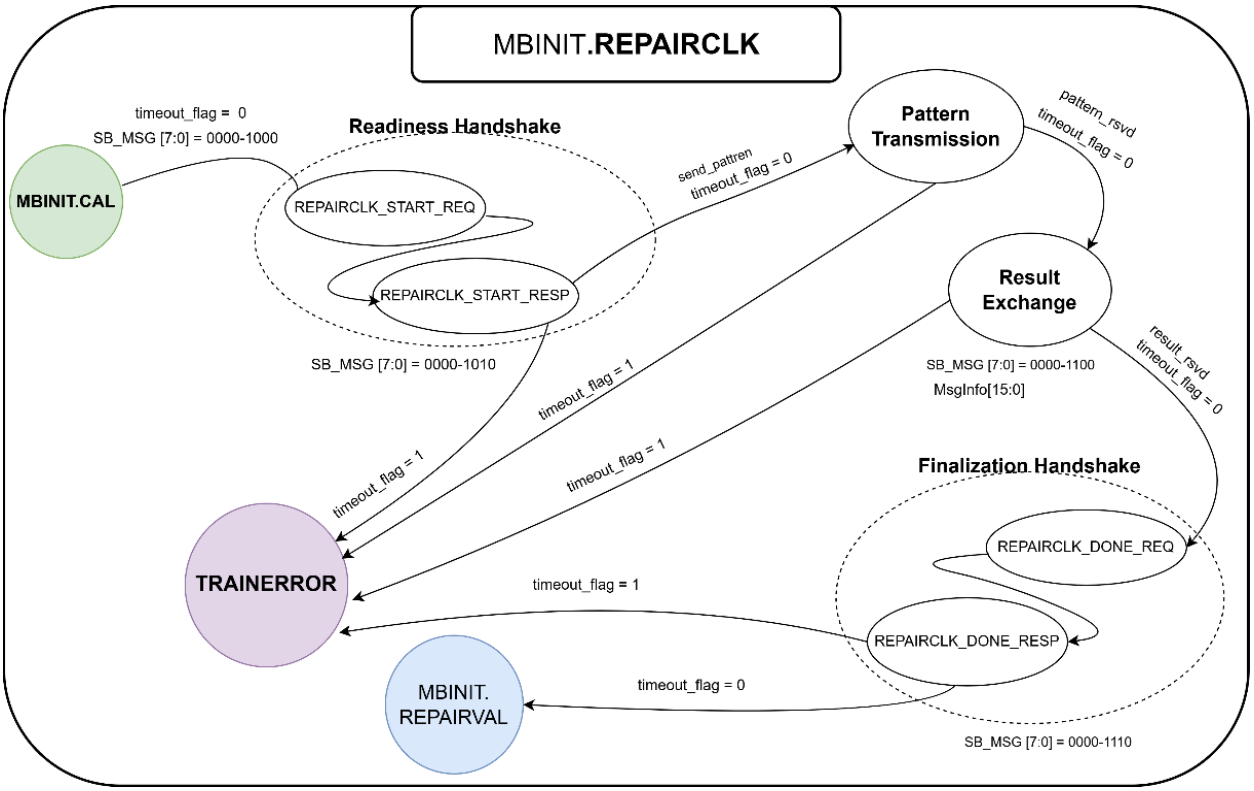


Figure 6-7: MBINIT.REPAIRCLK state diagram

MBINIT.REPAIRVAL

The REPAIRVAL sub-state validates the integrity of the valid lane. It drives the valid pattern transmitter (repairval_tx_pattern_en, repairval_tx_pattern_setup, repairval_tx_val_pattern_sel) and enables the valid comparator (repairval_rx_compare_en, repairval_rx_compare_setup) to measure valid framing reception. The sub-state then exchanges {MBINIT.REPAIRVAL} sideband messages with the remote die to share local results and receive the partner's result. Mutual pass confirms valid lane health and leads to repairval_done. A local or partner failure asserts repairval_error.

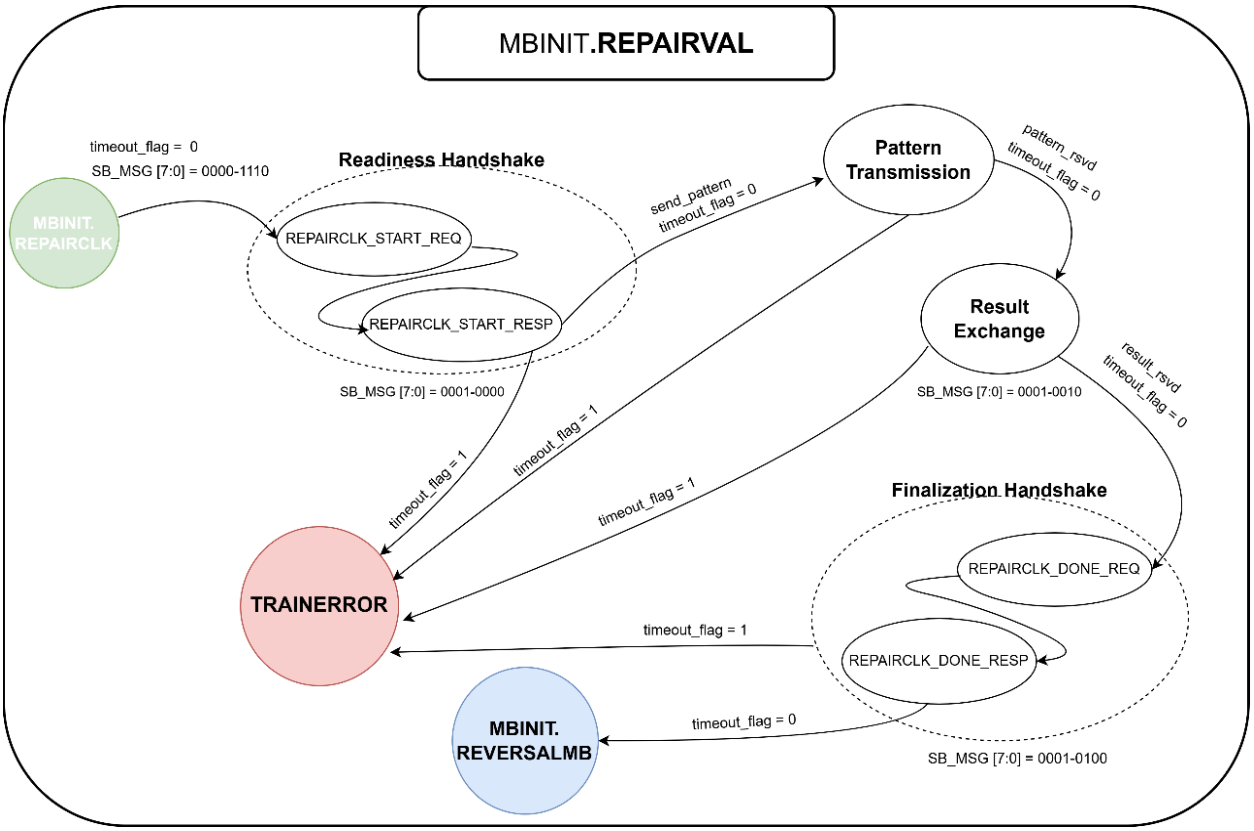


Figure 6-8: MBINIT.REPAIRVAL state diagram

Table 6-8: MBINIT.REPAIRVAL Interface Signals

Signal	Direction	Width	Description
repairval_init_req	Output	Enum	{init req}: readiness handshake.
repairval_init_resp	Input	Enum	{init resp}: partner clears prior comparison results.
repairval_result_req	Output	Enum	{result req}: requests Valid-lane compare result.
repairval_result_resp	Input	Enum	{result resp}: RVLD_L/RRDVLD_L compare bits.
repairval_apply_repair_req	Output	2	{apply repair req}: Valid-lane repair encoding (Advanced Package only).
repairval_apply_repair_resp	Input	Enum	{apply repair resp}: acknowledgment.
repairval_done_req	Output	Enum	{done req}: completion handshake.
repairval_done_resp	Input	Enum	{done resp}: closes the substate; transitions to REVERSALMB.
mb_tx_val_lane_sel	Output	2	Drives 128 iterations of the unscrambled VALTRAIN pattern with the forwarded clock.
mb_rx_val_lane_sel / mb_rx_clk_lane_sel	Output	1/1	Valid and clock receivers enabled to sample VALTRAIN.
mb_valid_done / mb_vcmp_pass	Input	1/1	Comparator completion/pass; failure exits to TRAINERROR (no Valid-lane redundancy in Standard Package).

MBINIT.REVERSALMB

The REVERSALMB sub-state detects and corrects physical lane reversal. It transmits per-lane ID patterns(reversalmb_tx_pattern_en,reversalmb_tx_pattern_setup,reversalmb_tx_data_pattern_sel) and enables the pattern comparator on the receive side. By comparing the received per-lane ID values against the expected identities, the sub-state determines whether the physical lane order is reversed relative to the logical assignment. If reversal is detected, it asserts mb_lane_reversal_req, which the MBINIT controller latches and forwards to the mainband datapath as the lane reversal enable. The sideband exchange (using reversalmb sideband messages) coordinates the detection outcome with the remote die. A clear_error_req is also asserted by this sub-state to reset the pattern comparator error accumulators before proceeding. On completion, reversalmb_done is asserted.

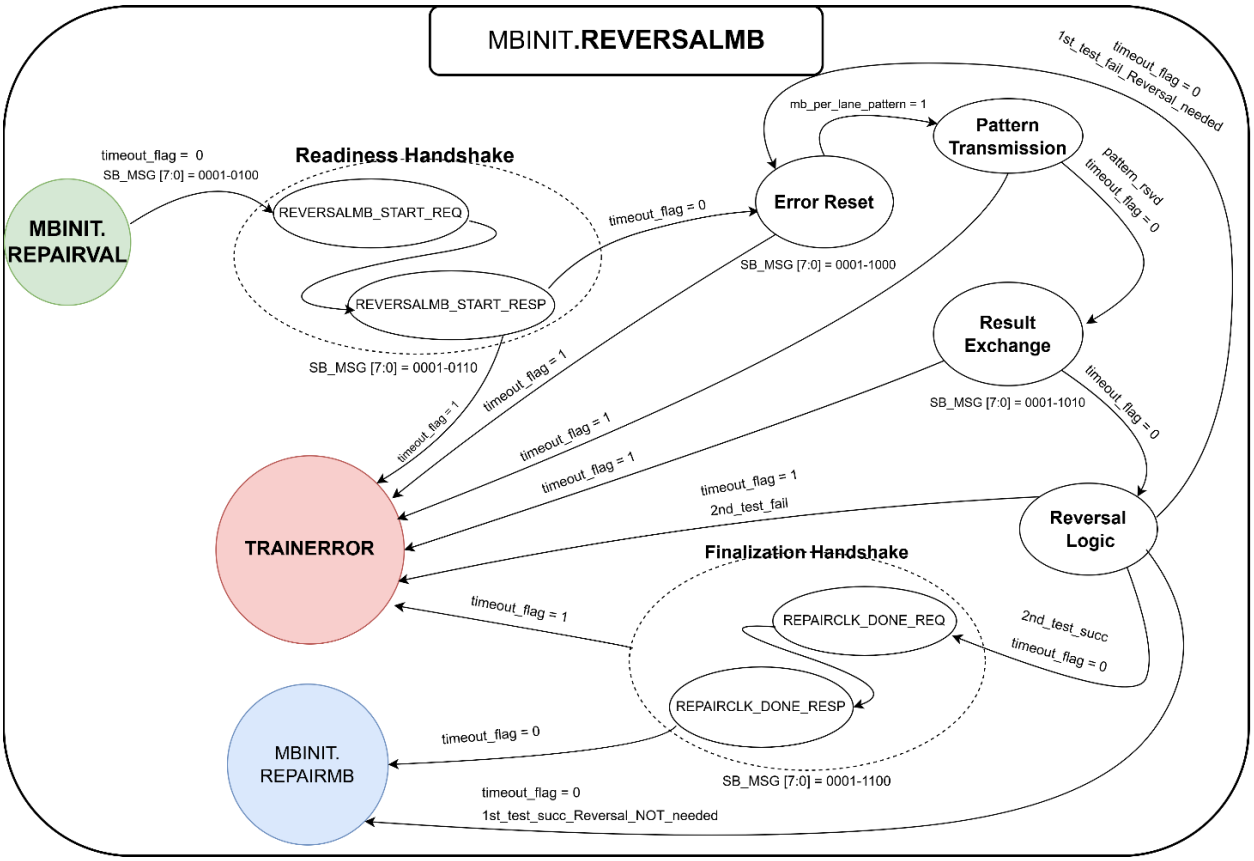


Figure 6-9: MBINIT.REVERSALMB state diagram

Table 6-9: MBINIT.REVERSALMB Interface Signals

Signal	Direction	Width	Description
reversalmb_init_req	Output	Enum	{init req}: readiness handshake before Per-Lane ID pattern transmission.
reversalmb_init_resp	Input	Enum	{init resp}: partner acknowledgment.
reversalmb_clear_error_req	Output	Enum	{clear error req}: clears prior results before a repeated attempt.
reversalmb_clear_error_resp	Input	Enum	{clear error resp}: acknowledgment.
reversalmb_result_req	Output	Enum	{result req}: requests exchange of per-lane detection results.
reversalmb_done_req	Output	Enum	{done req}: completion handshake.
reversalmb_done_resp	Input	Enum	{done resp}: closes the substate; transitions to REPAIRMB.
mb_tx_data_lane_sel	Output	2	Transmits the Per-Lane ID pattern across all active data lanes.
mb_rx_data_lane_sel	Output	1	Enables per-lane reception for reversal detection.
mb_reversal_en	Output	1	Applied on the transmitter if the majority criterion indicates a reversed connection.
mb_reversal_result	Input	16	Per-lane detection result vector; failure on both orientations exits to TRAINERROR.

MBINIT.REPAIRMB

The REPAIRMB sub-state executes the die-to-continuity point test (D2C PT) for all active data lanes and negotiates the final operational lane map. It uses the shared wrapper_D2C_PT block to run the D2C pattern sweep, receiving d2c_perlane_pass, d2c_aggr_pass, and d2c_val_pass results. Based on the per-lane pass vector, it determines whether the full-width (x16) lane set is functional or whether width degradation to x8 or x4 is required and feasible. The negotiation proceeds over sideband: the local die transmits its assessed lane map code, and the remote die responds with its own assessment. The agreed lane map code (the intersection that satisfies both dies) is then latched in mbinit_rx_data_lane_mask and mbinit_tx_data_lane_mask, which are forwarded to the LTSM wrapper as the operational lane configuration for all subsequent training and active operation. If no feasible lane set is found (all codes are 3'b000), repairmb_error is asserted.

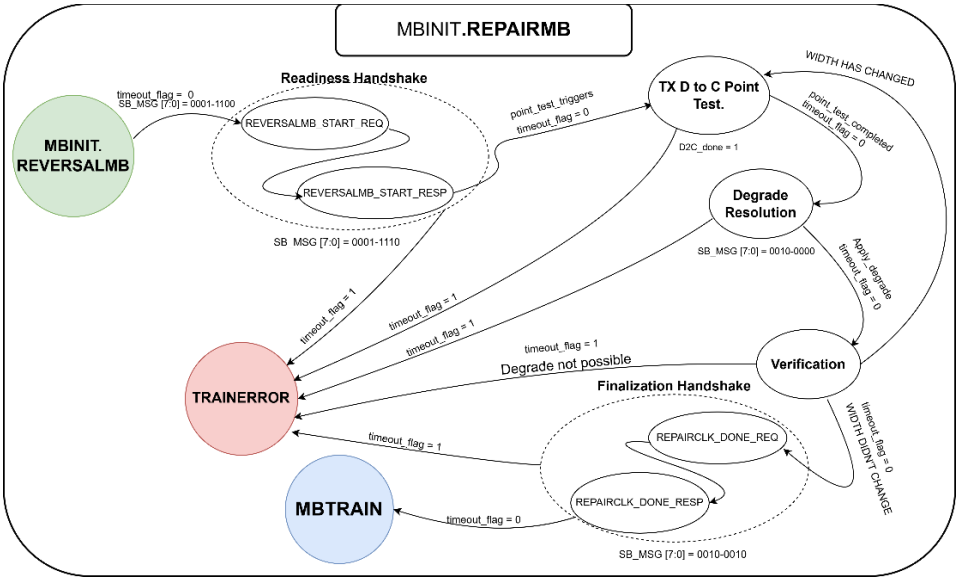


Figure 6-10: MBINIT.REAIRMB state diagram

Table 6-10: MBINIT.REPAIRMB Interface Signals

Signal	Direction	Width	Description
repairmb_start_req	Output	Enum	{start req}: readiness handshake before the data-to-clock point test.
repairmb_start_resp	Input	Enum	{start resp}: partner acknowledgment.
repairmb_apply_degrade_req	Output	3	{apply degrade req}: locally determined Standard Package logical lane-map encoding.
repairmb_apply_degrade_resp	Input	3	{apply degrade resp}: partner-reported lane-map encoding.
repairmb_end_req	Output	Enum	{end req}: completion handshake.
repairmb_end_resp	Input	Enum	{end resp}: closes MBINIT; transitions to MBTRAIN.
local_tx_pt_en / partner_tx_pt_en	Output	1/1	D2C point-test transmit enables at 4 GT/s.
d2c_pattern_setup	Output	3	Pattern-type select: 001b Data, 010b Valid, 100b Clock.
d2c_data_pattern_sel	Output	2	Data pattern select: LFSR, ID, or all-zero.
d2c_pattern_mode	Output	1	0: continuous; 1: burst.
mb_pcmp_per_lane_pass	Input	16	Per-lane comparison results used to determine the functional lane map.
mb_width_deg_tx / mb_width_deg_rx	Output	3/3	Configures the receiver to the agreed functional lane map.

6.2.6 MBTRAIN

MBTRAIN is implemented in wrapper_MBTRAIN, which integrates unit_MBTRAIN_ctrl (the 13-substate sequencer), 13 substate wrappers, and the shared D2C sweep engine. The MBTRAIN phase trains the mainband at the target operating data rate, centering clock and data phase interpolators, calibrating voltage references for valid and data lanes, performing receiver deskew, and confirming link speed stability. The controller follows a defined linear sequence with controlled loopback and multi-exit arcs at specific substates. Each substate wrapper internally combines local (initiator) and partner (responder) FSM done flags into a single done output presented to the controller.

MBTRAIN supports three alternative entry points in addition to the nominal VALVREF entry from MBINIT: SPEEDIDLE (from L1 exit or following a speed degrade request from LINKSPEED), TXSELFCAL (from PHYRETRAIN when no speed change is needed), and REPAIR (from PHYRETRAIN when width degradation is required). The entry point is selected by the controller based on the

mbtrain_speedidle_req, mbtrain_txselfcal_req, and mbtrain_repair_req inputs asserted at the moment mbtrain_en rises.

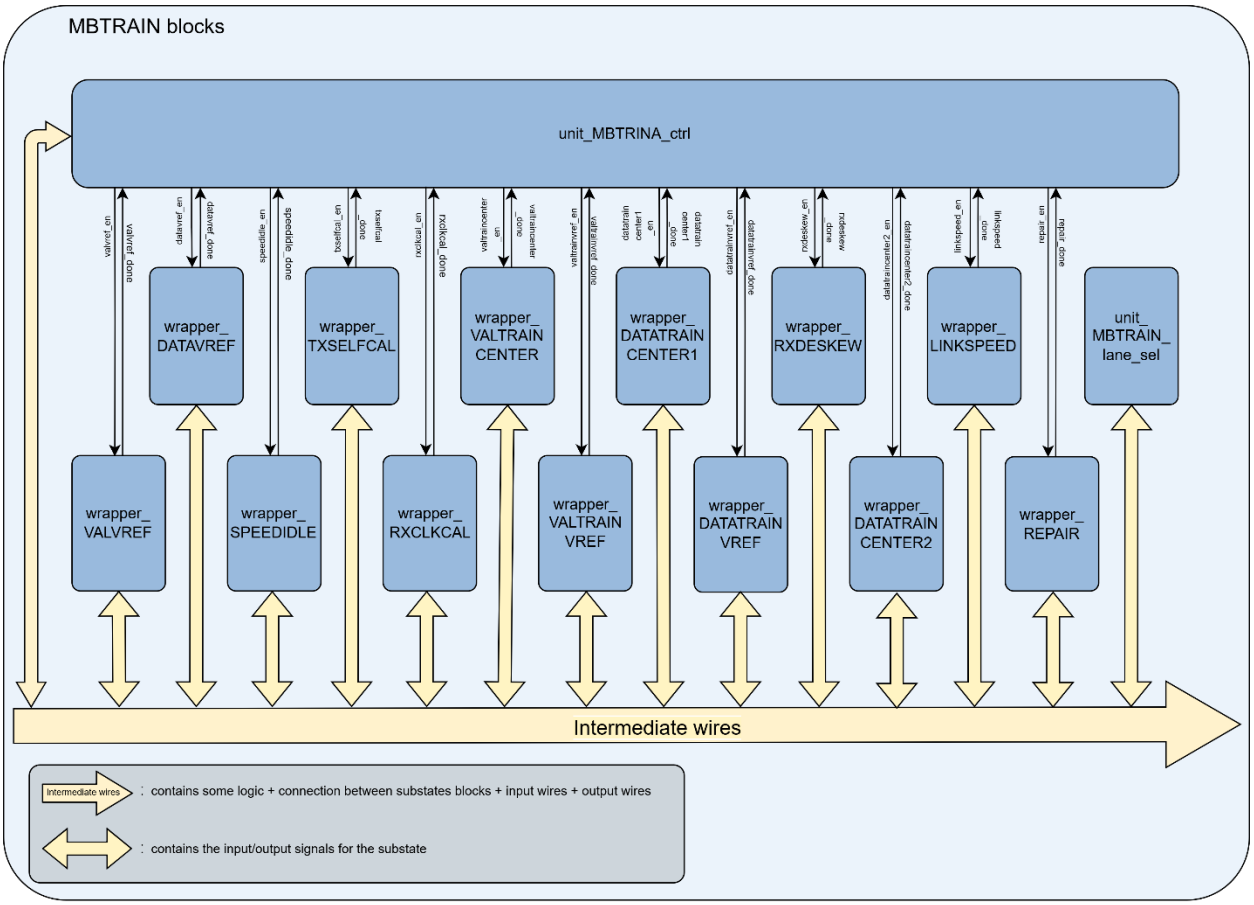


Figure 6-11: MBTRAIN Architecture

Table 6-11: MBTRAIN Control/Result Plane Overview

Signal	Direction	Width	Description
mbtrain_en	Input	1	Enable from top-level LTSM controller.
mbtrain_done	Output	1	Completion handshake to the LTSM controller.
current_mbtrain_substate	Output	Enum	Current MBTRAIN substate, logged to Reg_File Error Log.
mb_tx_clk/data/val/trk_lane_sel	Output	2 each	TX lane-behavior select: 00b low, 01b active, 1xb tri-state/electrical-idle; posture is substate-dependent.
mb_rx_clk/data/val/trk_lane_sel	Output	1 each	RX enable per group; posture is substate-dependent.
mb_tx_pattern_en / mb_tx_pattern_setup / mb_tx_clk_pattern_sel	Output	1/3/2	Pattern dispatch controls used principally during RXCLKCAL.

Signal	Direction	Width	Description
tx_sb_msg_valid / tx_sb_msg / tx_msginfo / tx_data_field	Output	1/8/16/64	Generic outbound sideband message bus shared by every substate FSM.
rx_sb_msg_valid / rx_sb_msg / rx_msginfo / rx_data_field	Input	1/8/16/64	Generic inbound sideband message bus.
phy_rx_valvref_ctrl / phy_rx_datavref_ctrl	Output	Enum each	Voltage-reference sweep control (VALVREF/VALTRAINVREF and DATAVREF/DATATRAINVREF).
phy_tx_val_pi_phase_ctrl / phy_tx_data_pi_phase_ctrl	Output	Enum each	Phase-interpolator centering control for Valid and Data transmit paths.
phy_rx_deskew_ctrl / phy_tx_eq_preset_ctrl	Output	Enum each	Receiver deskew and Tx equalization-preset control used by RXDESKEW.
phy_tx_selfcal_en	Output	1	Transmitter self-calibration enable used by TXSELFCAL.
phy_rx_clock_lock_en / phy_rx_track_lock_en / phy_rx_phase_detector_en	Output	1 each	Clock/track lock and phase-detector enables used by RXCLKCAL.
phy_tx_tckn_shift_en / phy_tx_tckn_shift / phy_tx_decrement_shift	Output	1 each	I/Q clock-phase shift control used by RXCLKCAL.
phy_rx_tckn_shift / phy_rx_decrement_shift / phy_tx_tckn_shift_out_of_range	Input	1 each	Shift feedback; out-of-range flag is a TRAINERROR cause in RXCLKCAL.
param_negotiated_max_speed	Input	4	Negotiated maximum common speed, carried from MBINIT into SPEEDIDLE.
phy_negotiated_speed	Output	4	Resolved operating speed applied during SPEEDIDLE.
mb_rx_data_lane_mask / mb_tx_data_lane_mask	Output	16/16	Lane-mask carry-through from MBINIT into MBTRAIN and REPAIR.

Table 6-12: MBTRAIN substates state diagram

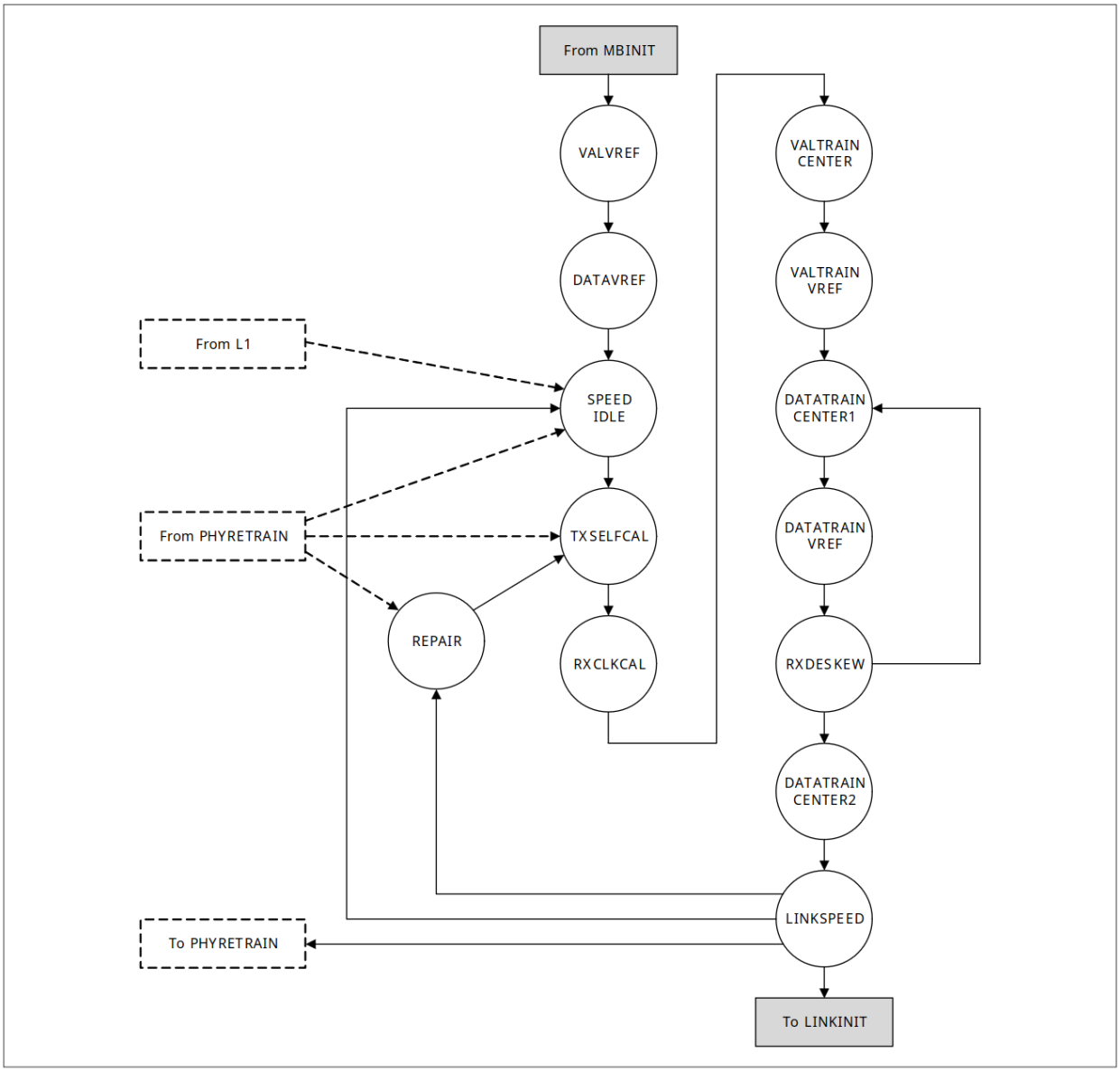


Figure 6-12: MBTRAIN state Diagram

MBTRAIN Controller

MBTRAIN Controller is the MBTRAIN sequencing FSM. It activates each of the 13 substates in turn, asserting a single enable per substate and waiting for the corresponding done before advancing. Two non-linear arcs are encoded. First, a loopback from RXDESKEW back to DATATRAINCENTER1: if RXDESKEW asserts dtc1_loopback_req, the controller returns to DATATRAINCENTER1 for a

second centering pass; this loopback may execute up to four times before the controller forces exit to DATATRAINCENTER2. Second, LINKSPEED has four possible exits: linkspeed_linkinit_req exits to MBTRAIN_DONE (→ LINKINIT), linkspeed_speedidle_req exits to SPEEDIDLE (speed degrade), linkspeed_repair_req exits to REPAIR (width degrade), and linkspeed_phyretrain_req exits the MBTRAIN domain (→ PHYRETRAIN at the LTSM level). The controller registers the linkspeed routing outputs to ensure clean handoff.

VALVREF

VALVREF performs receiver voltage reference calibration for the valid lane using a sweep of Vref codes at the negotiated initial data rate. The substate wrapper drives the D2C point test sweep, evaluating the valid lane eye for each code and converging on the optimal Vref setting. On completion, the settled Vref code is applied to the analog front-end via the training result interface and valvref_done is asserted.

Table 6-13: MBTRAIN.VALVREF Interface Signals

Signal	Direction	Width	Description
mb_tx_clk_lane_sel	Output	2	01b Active; forwarded clock driven unconditionally for the partner die's calibration.
mb_tx_data_lane_sel / mb_tx_trk_lane_sel	Output	2/2	00b; Data and Track transmitters held low.
mb_tx_val_lane_sel	Output	2	01b Active; partner drives the VALTRAIN pattern for local Vref sweep.
mb_rx_clk_lane_sel / mb_rx_val_lane_sel	Output	1/1	Enabled; local die samples VALTRAIN under the Vref sweep.
mb_rx_data_lane_sel / mb_rx_trk_lane_sel	Output	1/1	Disabled.
phy_rx_valvref_ctrl	Output	Enum	Sweeps the Rx Valid-lane voltage-reference code.
valvref_start_req / resp	Output/Input	Enum	Frames the sweep interval.
valvref_end_req / resp	Output/Input	Enum	Signals completion before transitioning to DATAVREF.

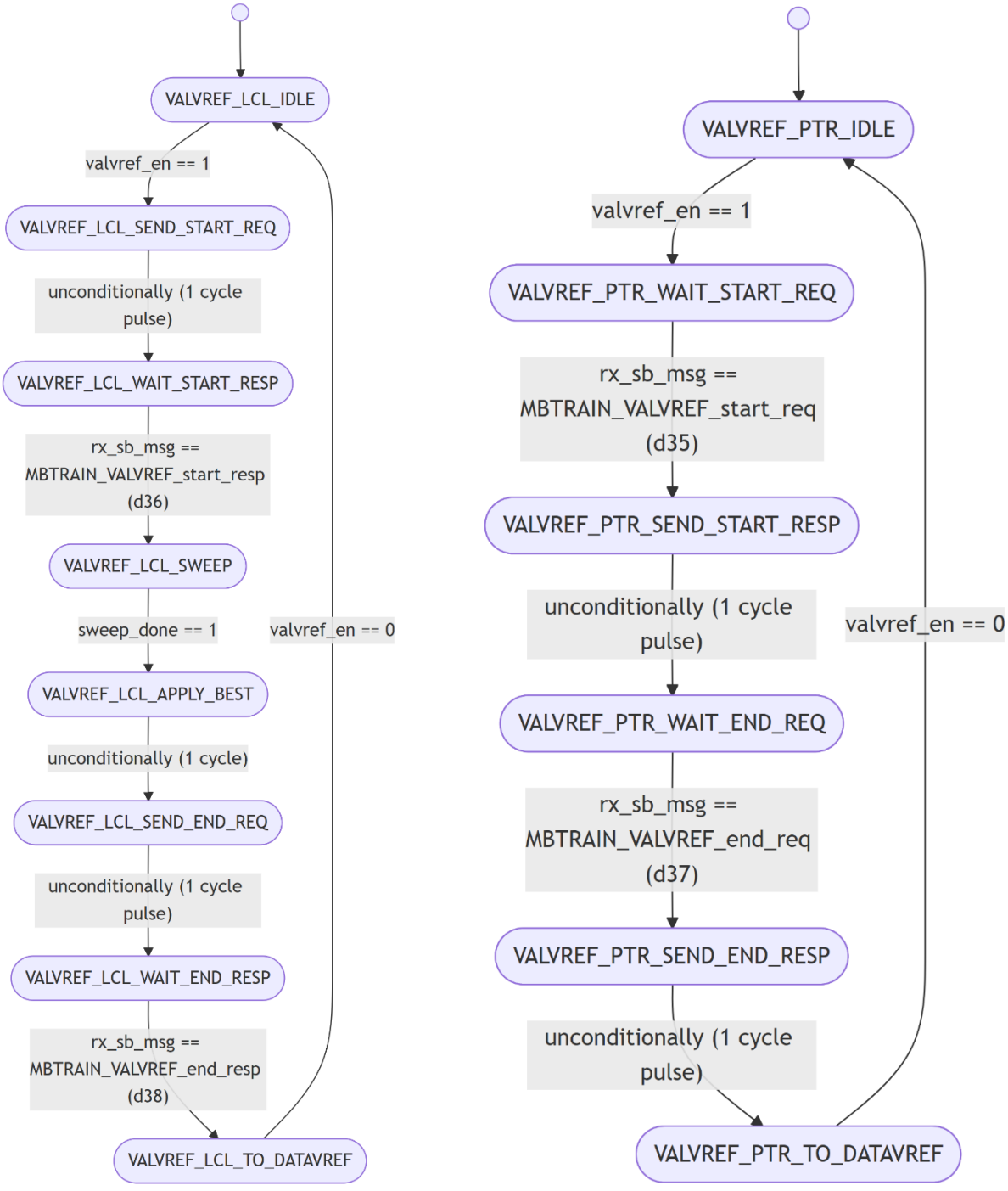


Figure 6-13: MBTAIN.VALVREV state diagram

DATAVREF

DATAVREF extends the Vref sweep to all active data lanes. The same D2C sweep infrastructure is applied across the full per-lane data eye to determine the optimal common or per-lane voltage reference. Once all active lanes have been calibrated, datavref_done is asserted and the controller advances to SPEEDIDLE.

Table 6-14: MBTRAIN.DATAVREF Interface Signals

Signal	Direction	Width	Description
mb_tx_clk_lane_sel	Output	2	01b Active; forwarded clock driven for the partner's calibration.
mb_tx_data_lane_sel / mb_tx_val_lane_sel / mb_tx_trk_lane_sel	Output	2 each	Held low; shared sweep engine overrides Data TX via partner_sweep_en.
mb_rx_clk_lane_sel / mb_rx_data_lane_sel / mb_rx_val_lane_sel	Output	1 each	Enabled to sample the sweep pattern.
mb_rx_trk_lane_sel	Output	1	Disabled.
phy_rx_datavref_ctrl	Output	Enum	Sweeps the Rx Data-lane voltage-reference code.
local_sweep_en / partner_sweep_en	Output	1/1	Coordinates the shared D2C sweep engine.
sweep_active_lanes / sweep_done	Input	16/1	Shared sweep engine status.
datavref_start_req / resp	Output/Input	Enum	Frames the sweep interval.
datavref_end_req / resp	Output/Input	Enum	Completion; transitions to SPEEDIDLE.

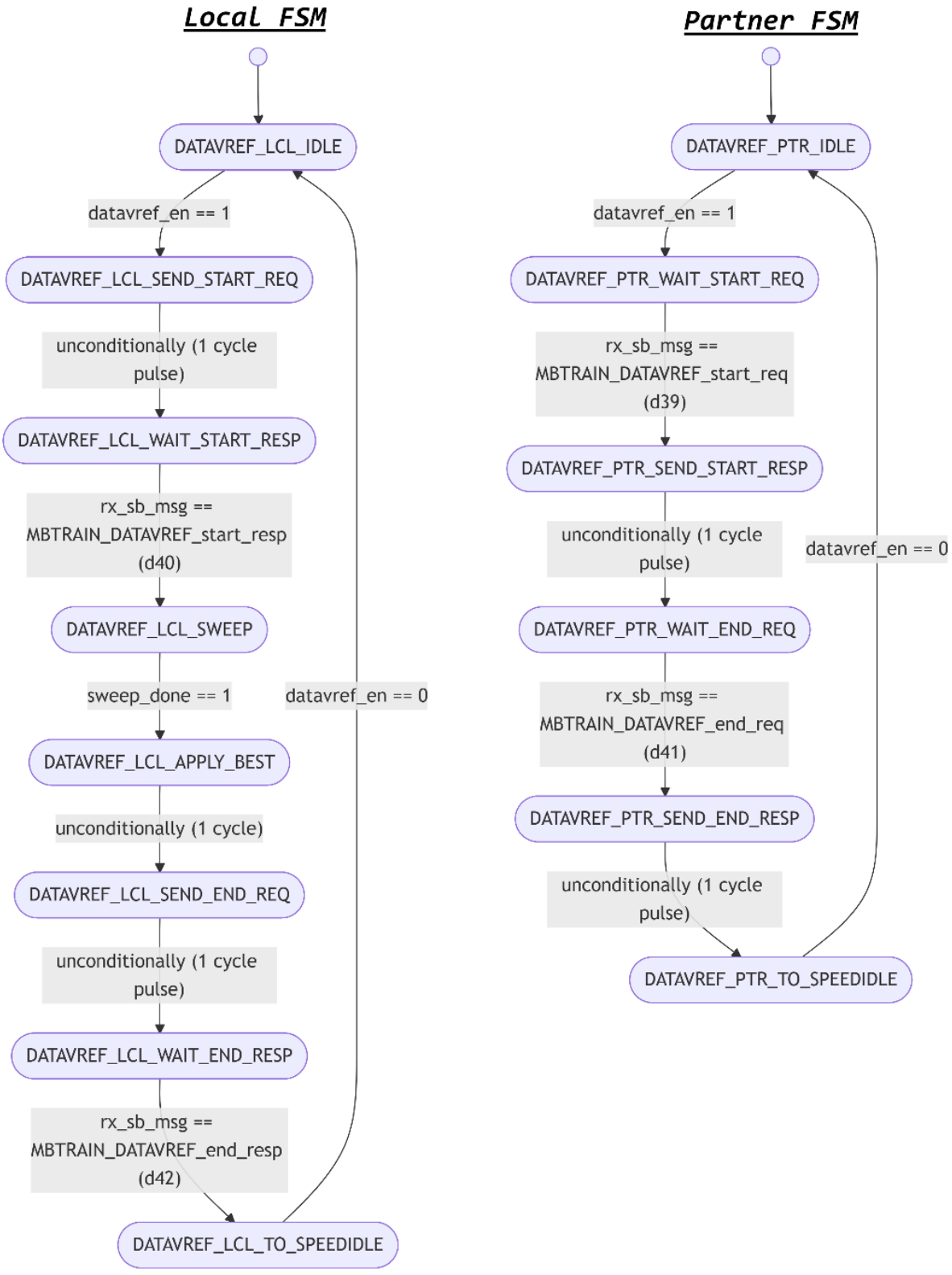


Figure 6-14: MBTRAIN.DATAVREF state diagram

SPEEDIDLE

SPEEDIDLE negotiates the link operating speed between the two dies and transitions the mainband from the 4 GT/s initialization rate to the agreed target data rate. Both dies advertise their maximum supported speed through the MBINIT.PARAM exchange; SPEEDIDLE enforces the common minimum. The sub-state manages the PLL speed selection (mb_pll_speed_sel) and any associated electrical idle periods required during the rate change. If the negotiated speed cannot be achieved, speedidle asserts its error output and the controller routes to TRAINERROR. SPEEDIDLE is re-entered on each speed degrade iteration from LINKSPEED and on every L1 exit, allowing the link to recover from transient instability without a full cold restart.

Table 6-15: MBTRAIN.SPEEDIDLE Interface Signals

Signal	Direction	Width	Description
mb_tx_clk_lane_sel	Output	2	Active; forwarded clock kept active during speed settling.
mb_tx_data_lane_sel / mb_tx_val_lane_sel / mb_tx_trk_lane_sel	Output	2 each	Held low.
mb_rx_clk_lane_sel	Output	1	Enabled to track the settling forwarded clock.
mb_rx_data_lane_sel / mb_rx_val_lane_sel / mb_rx_trk_lane_sel	Output	1 each	Disabled.
param_negotiated_max_speed	Input	4	Negotiated (or fallback-degraded) rate from MBINIT/LINKSPEED.
phy_negotiated_speed	Output	4	Applied operating rate; PLL clock-rate change occurs here.
speedidle_req	Input	1	Re-entry select: asserted on L1 exit and on PHYRETRAIN/LINKSPEED speed-degrade re-entry.
speedidle_speed_nego (sideband exchange)	Output/Input	Enum	Confirms the resolved common data rate; failure with no lower rate exits to TRAINERROR.

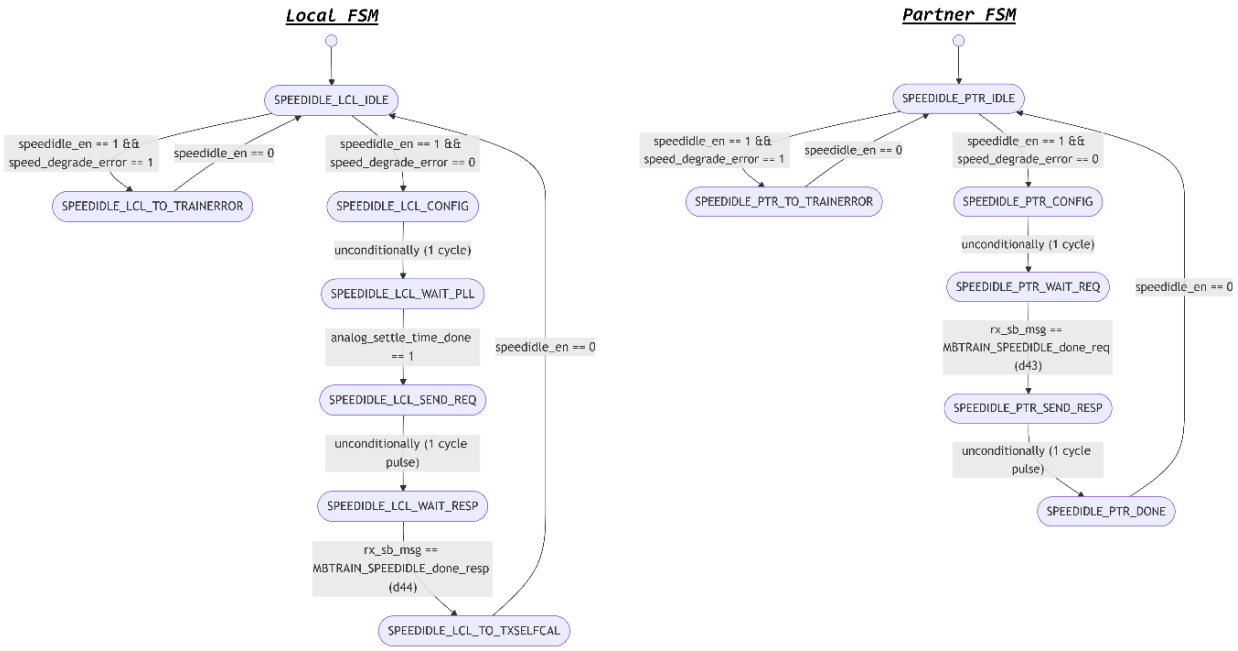


Figure 6-15: MBTRAIN.SPEEDIDLE state diagram

TXSELFCAL

TXSELFCAL performs transmitter self-calibration at the negotiated operating data rate. It drives the TX self-calibration sequence, which measures and corrects any offset or timing skew internal to the transmitter circuitry. No sideband message exchange with the remote die is required for TXSELFCAL; the calibration is performed locally and results are applied through the analog control interface. On completion, txselfcal_done is asserted.

Table 6-16: MBTRAIN.TXSELFCAL Interface Signals

Signal	Direction	Width	Description
mb_tx_clk/data/val/trk_lane_sel	Output	2 each	10b Tri-state (Hi-Z) on all four transmitter groups.
mb_rx_clk/data/val/trk_lane_sel	Output	1 each	Permitted to be disabled.
phy_tx_selfcal_en	Output	1	Enables clock and data transmitter self-calibration.
txselfcal_req	Input	1	Re-entry select: asserted on PHYRETRAIN re-entry when no lane repair is required.
txselfcal_done_handshake	Output/Input	Enum	Signals local self-calibration completion before proceeding to RXCLKCAL.

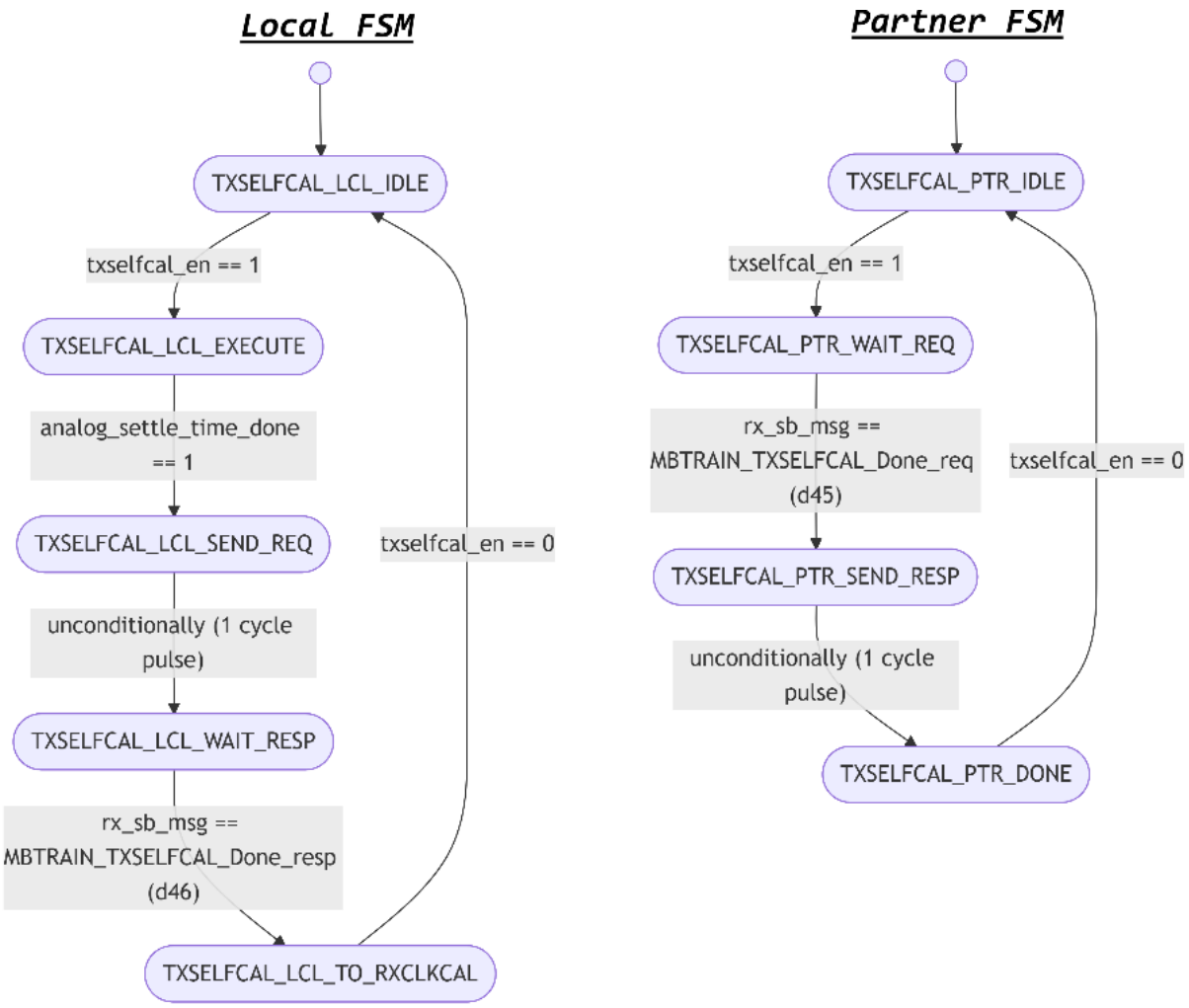


Figure 6-16: MBTRAIN.TXSELFCAL state diagram

RXCLKCAL

RXCLKCAL performs receiver clock calibration including I/Q phase locking. It enables the RX clock calibration sequence, during which the receiver phase interpolator is swept to align the sampling clock to the center of the incoming data eye. An IQ partner sub-block coordinates with the transmitter side to exercise both phases of the differential clock. If the phase interpolator reaches its sweep limit without converging, RXCLKCAL asserts an error and the controller routes to TRAINERROR. On successful lock, rxclkcal_done is asserted.

Table 6-17: MBTRAIN.RXCLKCAL Interface Signals

Signal	Direction	Width	Description
mb_tx_clk_lane_sel / mb_tx_trk_lane_sel	Output	2/2	Active when tx_clk_active is set (partner FSM transmits clock/track); held low otherwise.
mb_rx_clk_lane_sel / mb_rx_trk_lane_sel	Output	1/1	Enabled when rx_clk_active is set (LOCAL FSM receiving remote clock); disabled otherwise.
mb_tx_pattern_en / mb_tx_pattern_setup / mb_tx_clk_pattern_sel	Output	1/3/2	Dispatches the clock training pattern for I/Q lock correction.
phy_rx_clock_lock_en / phy_rx_track_lock_en / phy_rx_phase_detector_en	Output	1 each	Enables clock/track lock loops and the phase detector.
phy_tx_tckn_shift_en / phy_tx_tckn_shift / phy_tx_decrement_shift	Output	1 each	Drives incremental I/Q phase-shift corrections on TCKN.
phy_rx_tckn_shift / phy_rx_decrement_shift	Input	1/1	Shift feedback.
phy_tx_tckn_shift_out_of_range	Input	1	Exits to TRAINERROR when the residual phase shift exceeds the TCKN driver limit.
rxclkcal_done_handshake	Output/Input	Enum	Signals I/Q lock completion before proceeding to VALTRAINCENTER.

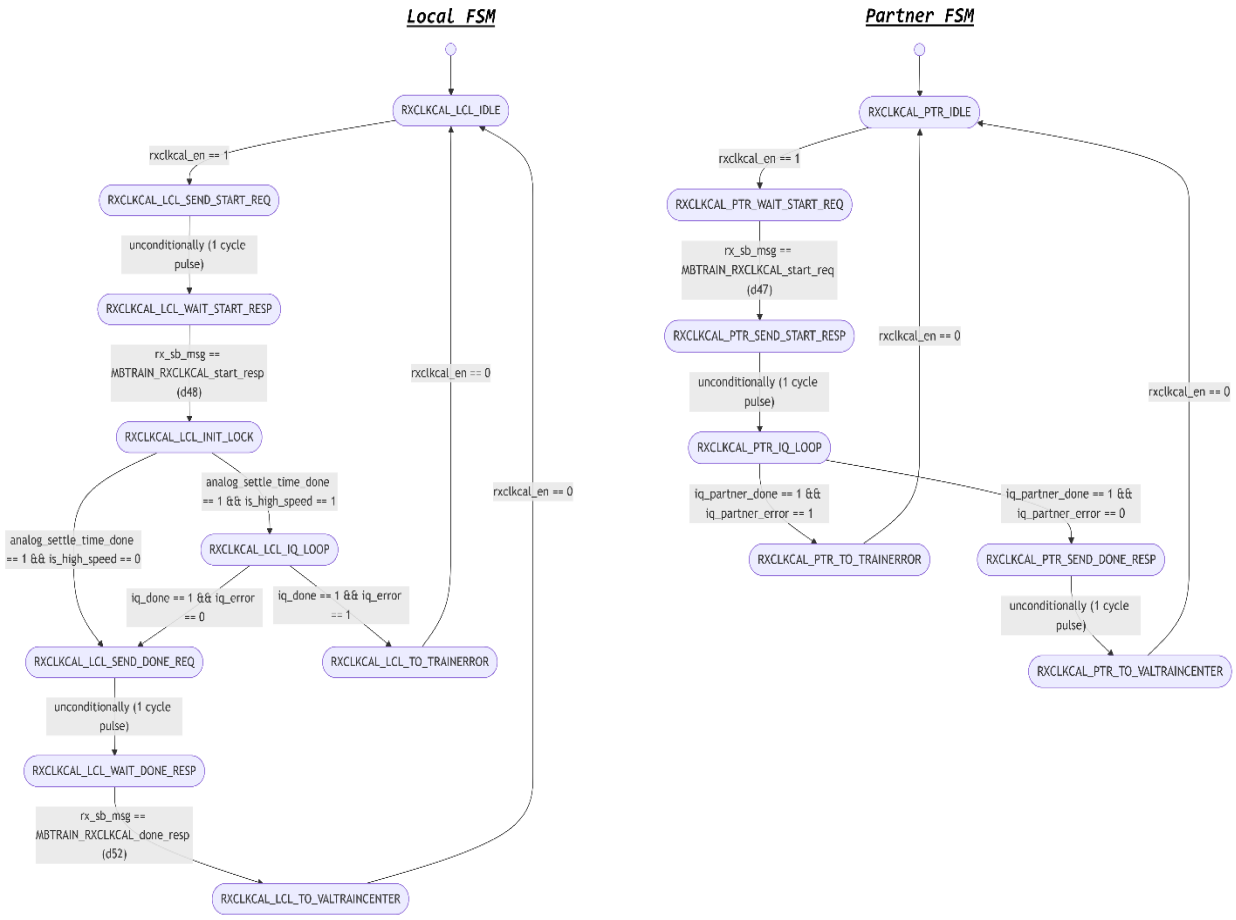


Figure 6-17: MBTARIN.RXCLKCAL state diagram

VALTRAINCENTER

VALTRAINCENTER centers the transmit phase interpolator for the valid lane by running a D2C sweep at the negotiated operating speed. The local die sweeps the TX PI code for the valid lane while the partner die measures the resulting eye. The sweep engine (wrapper_D2C_PT, instantiated at the LTSM wrapper level and shared across substates) returns the optimal code, which is applied before valtraincenter_done is asserted.

Table 6-18: MBTRAIN.VALTRAINCENTER Interface Signals

Signal	Direction	Width	Description
mb_tx_clk_lane_sel	Output	2	01b if high-speed/continuous-clock, else 00b.
mb_tx_data_lane_sel / mb_tx_trk_lane_sel	Output	2/2	Held low.
mb_tx_val_lane_sel	Output	2	01b Active; partner drives the VALTRAIN pattern.
mb_rx_clk_lane_sel / mb_rx_val_lane_sel	Output	1/1	Enabled.
mb_rx_data_lane_sel / mb_rx_trk_lane_sel	Output	1/1	Disabled.
phy_tx_val_pi_phase_ctrl	Output	Enum	Centers the transmit phase interpolator for the Valid lane.
valtraincenter_start_req / resp, end_req / resp	Output/Input	Enum	Frames the centering interval; transitions to VALTRAINVREF.

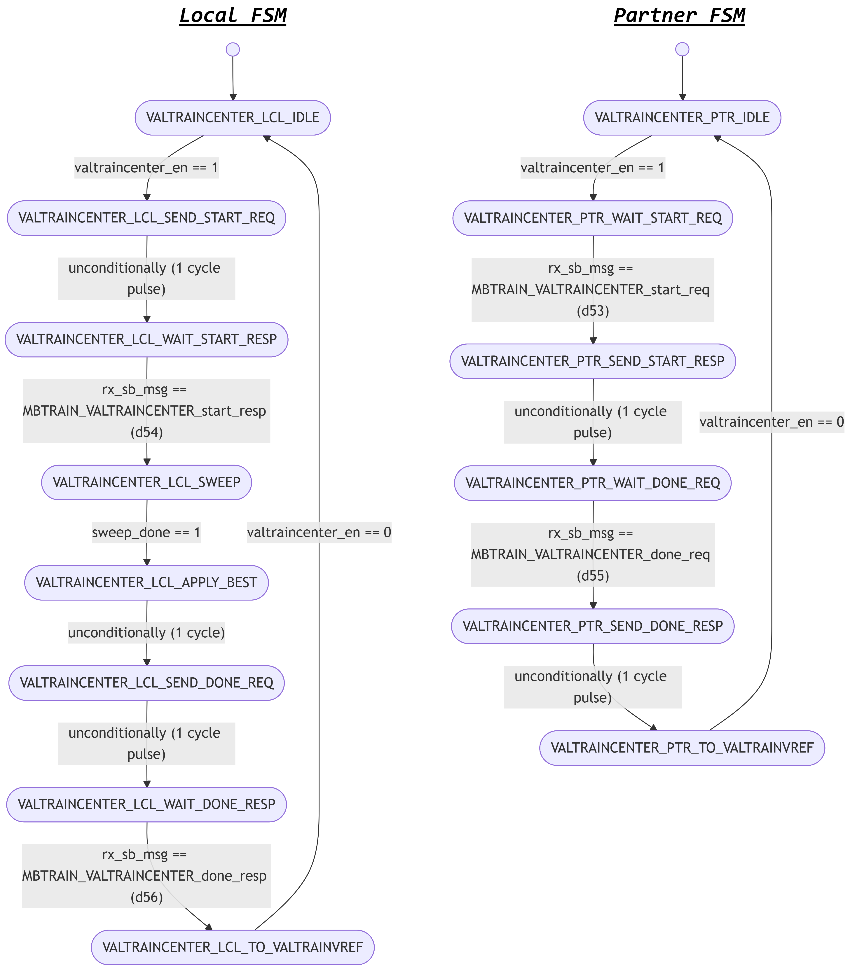


Figure 6-18: MBTRAIN.VALTRAINCENTER state diagram

VALTRAINVREF

VALTRAINVREF trains the receive voltage reference for the valid lane at the operating speed, complementing the VALVREF calibration performed at 4 GT/s. The D2C sweep evaluates the valid lane eye at the target rate and settles a refined Vref code. On completion valtrainvref_done is asserted and the controller advances to DATATRAINCENTER1.

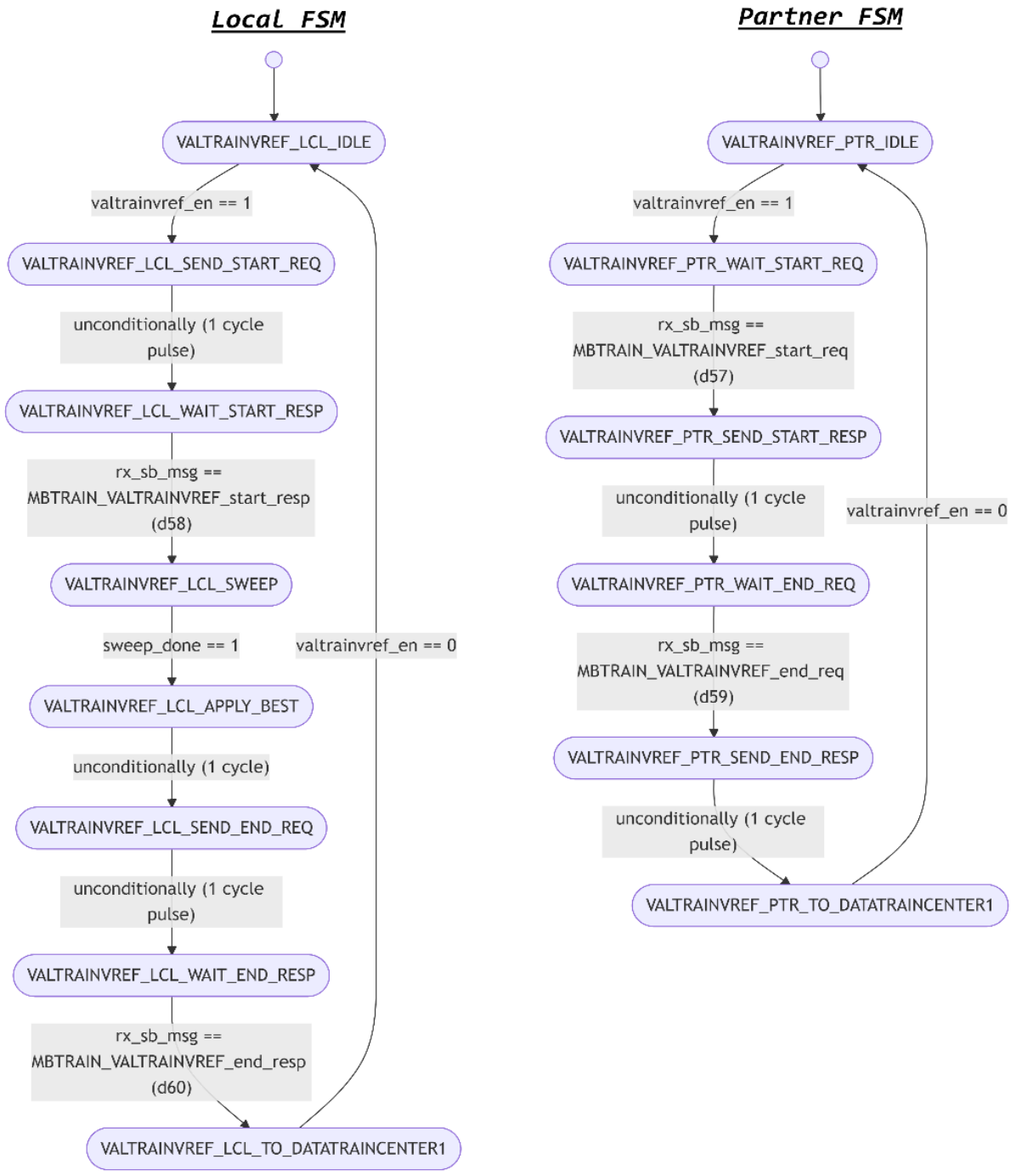


Figure 6-19: MBTRAIN.VALTRAINVREF state diagram

Table 6-19: MBTRAIN.VALTRAINVREF Interface Signals

Signal	Direction	Width	Description
mb_tx_clk_lane_sel	Output	2	01b if high-speed/continuous-clock, else 00b.
mb_tx_val_lane_sel	Output	2	01b Active; partner drives VALTRAIN.
mb_tx_data_lane_sel / mb_tx_trk_lane_sel	Output	2/2	Held low.
mb_rx_clk_lane_sel / mb_rx_val_lane_sel	Output	1/1	Enabled.
mb_rx_data_lane_sel / mb_rx_trk_lane_sel	Output	1/1	Disabled.
phy_rx_valvref_ctrl	Output	Enum	Rx Valid-lane Vref training, shared control path with VALVREF.
valtrainvref_start_req / resp, end_req / resp	Output/Input	Enum	Frames the sweep; transitions to DATATRAINCENTER1.

DATATRAINCENTER1

DATATRAINCENTER1 performs the first pass of transmit data PI centering across all active data lanes using the shared D2C sweep engine. Both local and partner sub-FSMs collaborate: the local die sweeps the TX PI codes while the partner measures the resulting data eye. The best PI code per lane is determined and applied. On completion, dtc1_done is asserted. If RXDESKEW subsequently asserts dtc1_loopback_req, the controller returns here for a refined second centering pass.

Table 6-20: MBTRAIN.DATATRAINCENTER1 Interface Signals

Signal	Direction	Width	Description
mb_tx_clk_lane_sel	Output	2	01b if high-speed/continuous-clock, else 00b.
mb_tx_data_lane_sel / mb_tx_val_lane_sel / mb_tx_trk_lane_sel	Output	2 each	Held low; sweep engine overrides Data TX.
mb_rx_clk_lane_sel / mb_rx_data_lane_sel / mb_rx_val_lane_sel	Output	1 each	Enabled.
mb_rx_trk_lane_sel	Output	1	Disabled.
phy_tx_data_pi_phase_ctrl	Output	Enum	Transmit PI centering for Data lanes, Pass 1 (shared with DATATRAINCENTER2).
dtc1_loopback_req	Input	1	Re-entry from RXDESKEW; up to 4 loopback iterations if the eye-width target is not met.
dtc1_start_req / resp, end_req / resp	Output/Input	Enum	Frames the centering interval; transitions to DATATRAINVREF.

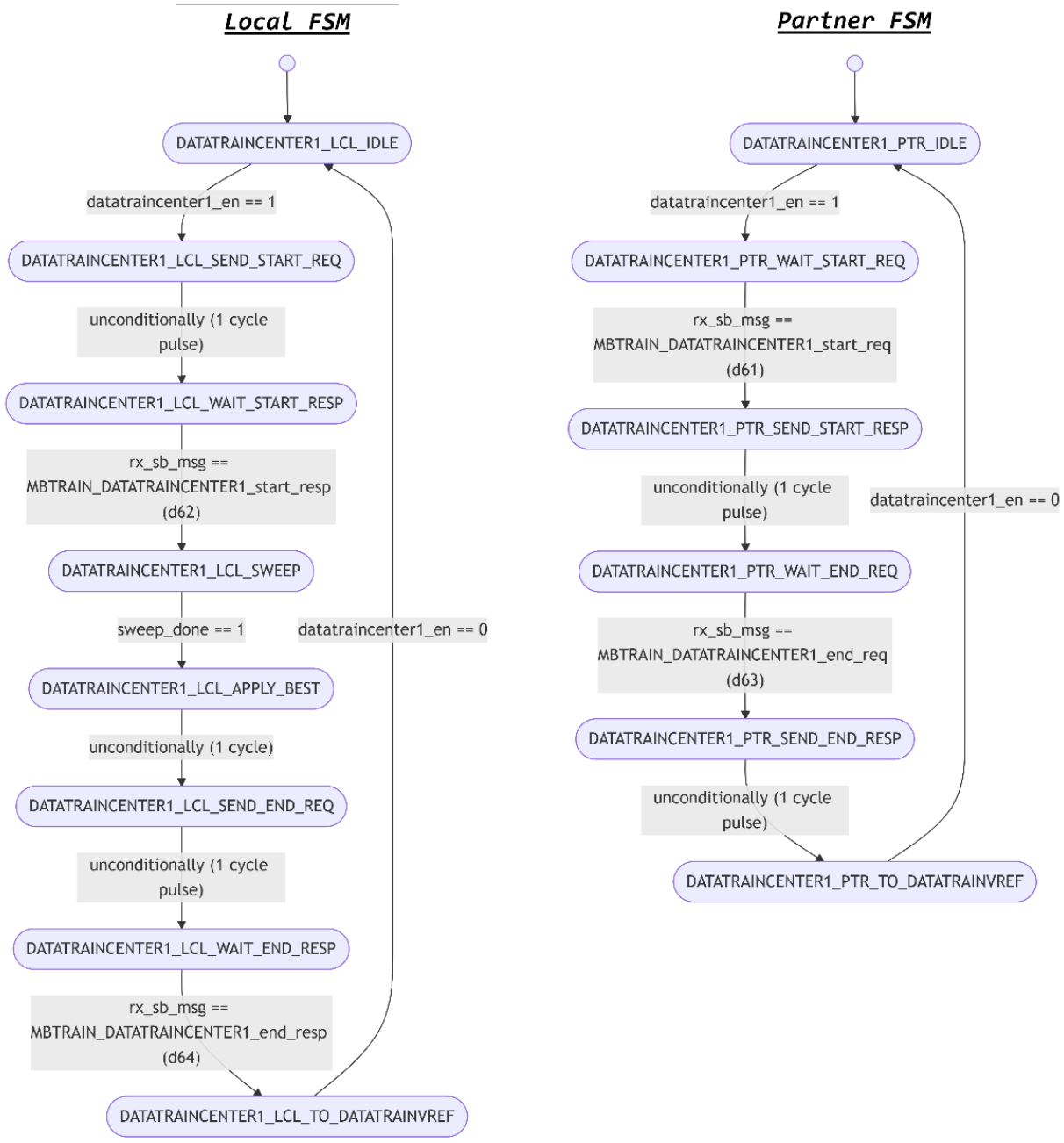


Figure 6-20: MBTRAIN.DATATRAINCENTER1

DATATRAINVREF

DATATRAINVREF trains the receive voltage reference for all active data lanes at the operating speed. The D2C sweep is applied across all lanes in the active set, refining the Vref codes established during DATAVREF at 4 GT/s. On completion, datatrainvref_done is asserted.

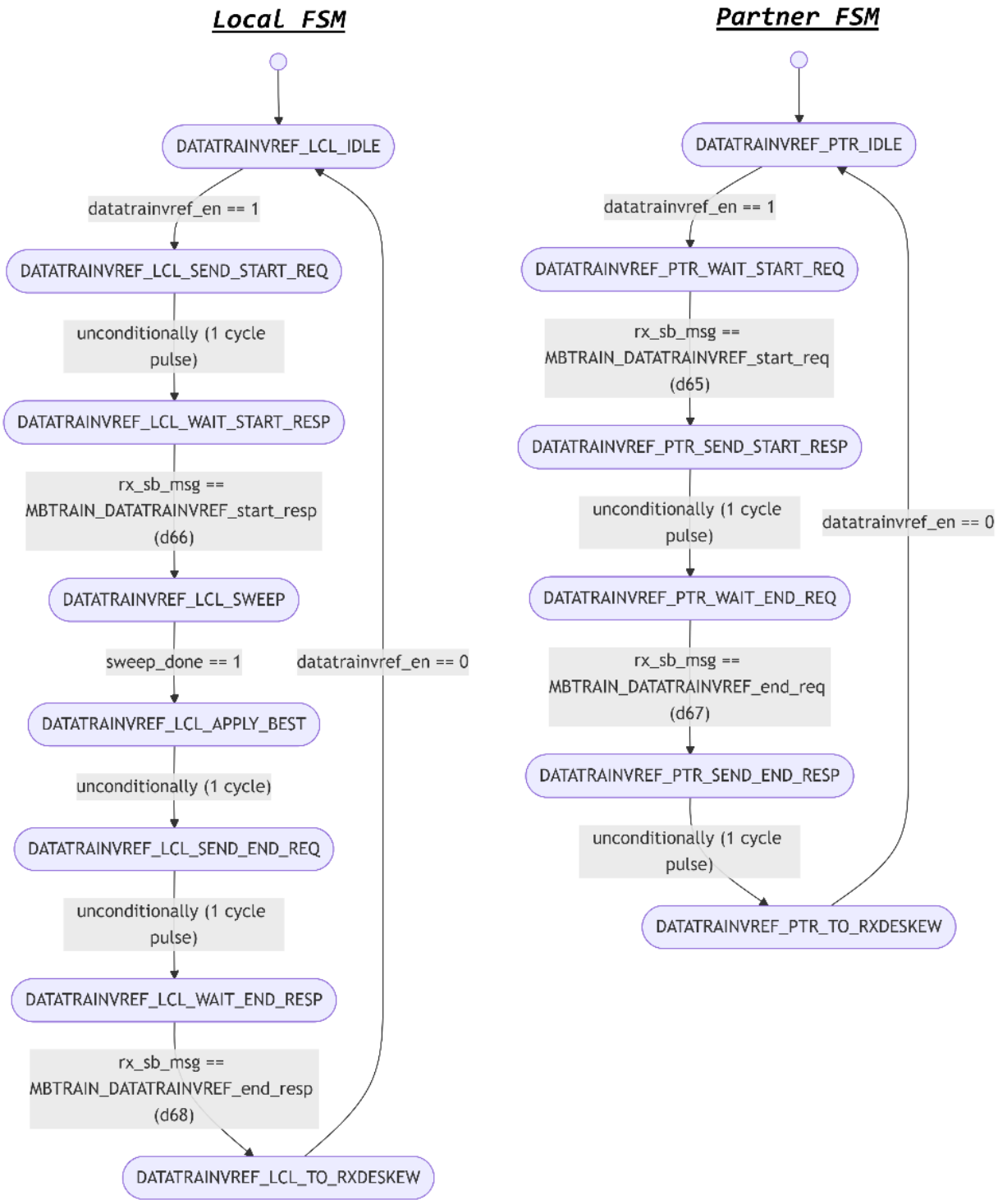


Figure 6-21: DATATRAINVREF state diagram

Table 6-21: MBTRAIN.DATATRAINVREF Interface Signals

Signal	Direction	Width	Description
mb_tx_clk_lane_sel	Output	2	01b if high-speed/continuous-clock, else 00b.
mb_tx_data_lane_sel / mb_tx_val_lane_sel / mb_tx_trk_lane_sel	Output	2 each	Held low; sweep engine active.
mb_rx_clk_lane_sel / mb_rx_data_lane_sel / mb_rx_val_lane_sel	Output	1 each	Enabled.
mb_rx_trk_lane_sel	Output	1	Disabled.
phy_rx_datavref_ctrl	Output	Enum	Rx Data-lane Vref training, shared control path with DATAVREF.
best_code[0:15] / swept_code / sweep_done	Input	16×Enum/Enum/1	Per-lane best-midpoint sweep results.
datatrainvref_start_req / resp, end_req / resp	Output/Input	Enum	Frames the sweep; transitions to RXDESKEW.

RXDESKEW

RXDESKEW measures the receive data eye width across all active lanes and applies equalization preset adjustments to correct inter-lane skew. If the measured eye width is insufficient and the DTC1 loop count has not yet reached four, rxdeskew asserts dtc1_loopback_req, causing the controller to return to DATATRAINCENTER1 for a refined centering pass. If the loop count is exhausted or eye width is adequate, the controller advances to DATATRAINCENTER2. RXDESKEW is active only at high-speed operating modes; at lower data rates it asserts rxdeskew_done immediately.

Table 6-22: MBTRAIN.RXDESKEW Interface Signals

Signal	Direction	Width	Description
mb_tx_clk_lane_sel	Output	2	01b if high-speed/continuous-clock, else 00b; substate active only in high-speed mode.
mb_tx_data_lane_sel / mb_tx_val_lane_sel / mb_tx_trk_lane_sel	Output	2 each	Held low.
mb_rx_clk_lane_sel / mb_rx_data_lane_sel / mb_rx_val_lane_sel	Output	1 each	Enabled to measure per-lane eye width.
mb_rx_trk_lane_sel	Output	1	Disabled.
phy_rx_deskew_ctrl / phy_tx_eq_preset_ctrl	Output	Enum each	Per-lane deskew adjustment and Tx equalization-preset tuning.

Signal	Direction	Width	Description
current_ltsm_state	Input	Enum	Monitored to detect an external RESET during the deskew loop.
dtc1_loopback_req	Output	1	Requests re-entry to DATATRAINCENTER1 (up to 4 iterations); exceeding the limit exits to TRAINERROR.
rxdeskew_start_req / resp, end_req / resp	Output/Input	Enum	Frames the deskew/EQ-preset tuning interval; transitions to DATATRAINCENTER2.

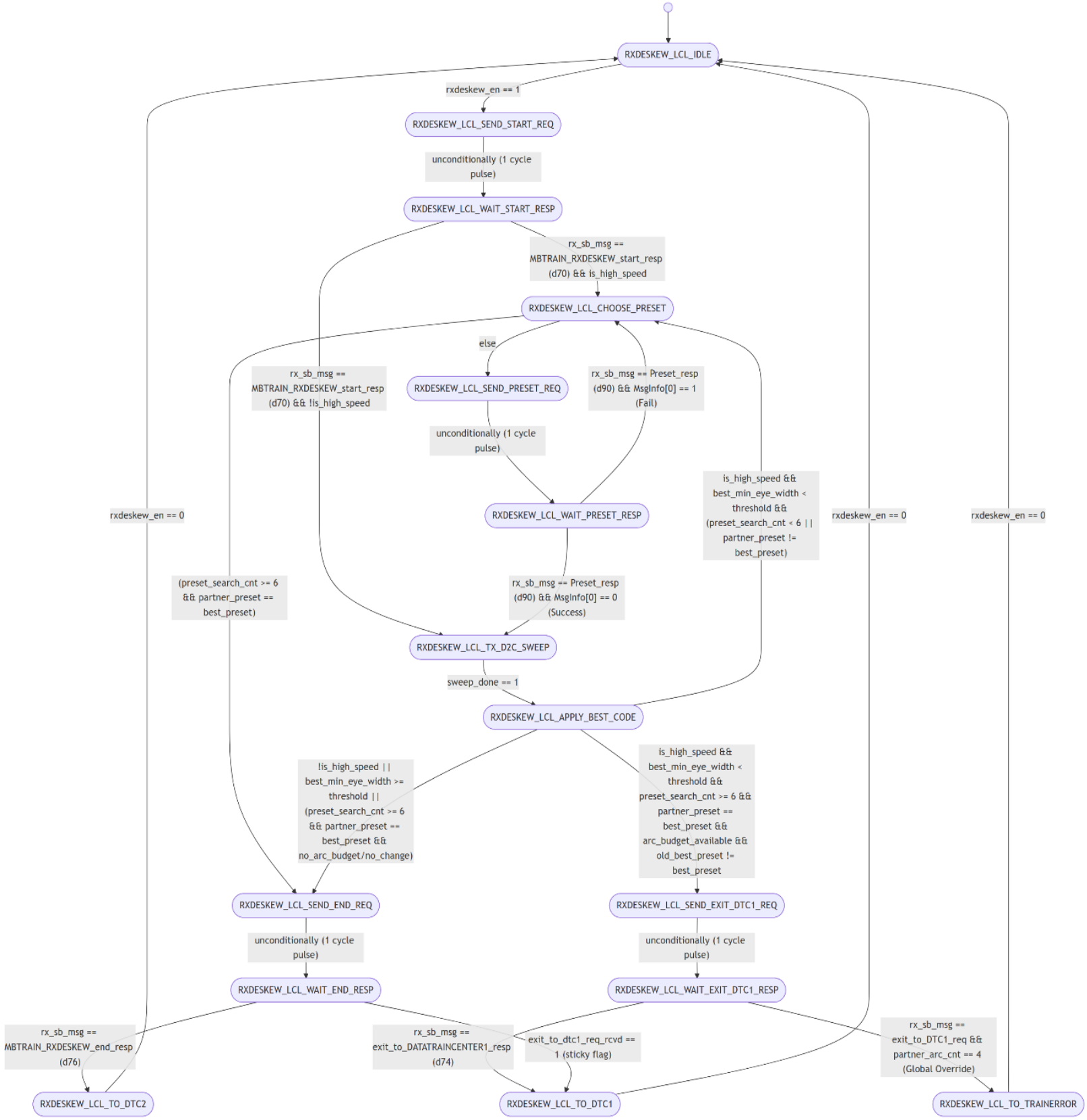

RXDESKEW_LCL_IDLE
rxdeskew_en == 1
RXDESKEW_LCL_SEND_START_REQ
unconditionally (1 cycle pulse)
RXDESKEW_LCL_WAIT_START_RESP
rx_sb_msg == MBTRAIN_RXDESKEW_start_resp (d70) && is_high_speed
RXDESKEW_LCL_CHOOSE_PRESET
else
rx_sb_msg == MBTRAIN_RXDESKEW_start_resp (d70) && !is_high_speed
RXDESKEW_LCL_SEND_PRESET_REQ
rx_sb_msg == Preset_resp (d90) && MsgInfo[0] == 1 (Fail)
unconditionally (1 cycle pulse)
RXDESKEW_LCL_WAIT_PRESET_RESP
rxdeskew_en == 0
is_high_speed && best_min_eye_width < threshold && (preset_search_cnt < 6 || partner_preset != best_preset)
rxdeskew_en == 0
rxdeskew_en == 0
(preset_search_cnt >= 6 && partner_preset == best_preset)
rx_sb_msg == Preset_resp (d90) && MsgInfo[0] == 0 (Success)
RXDESKEW_LCL_TX_D2C_SWEEP
sweep_done == 1
RXDESKEW_LCL_APPLY_BEST_CODE
!is_high_speed || best_min_eye_width >= threshold || (preset_search_cnt >= 6 && partner_preset == best_preset && no_arc_budget/no_change)
is_high_speed && best_min_eye_width < threshold && preset_search_cnt >= 6 && partner_preset == best_preset && arc_budget_available && old_best_preset != best_preset
RXDESKEW_LCL_SEND_END_REQ
RXDESKEW_LCL_SEND_EXIT_DTC1_REQ
unconditionally (1 cycle pulse)
unconditionally (1 cycle pulse)
RXDESKEW_LCL_WAIT_END_RESP
RXDESKEW_LCL_WAIT_EXIT_DTC1_RESP
rx_sb_msg == MBTRAIN_RXDESKEW_end_resp (d76)
rx_sb_msg == exit_to_DATATRAINCENTER1_resp (d74)
exit_to_dtc1_req_rcvd == 1 (sticky flag)
rx_sb_msg == exit_to_DTC1_req && partner_arc_cnt == 4 (Global Override)
RXDESKEW_LCL_TO_DTC2
RXDESKEW_LCL_TO_DTC1
RXDESKEW_LCL_TO_TRAINERROR

Figure 6-22: MBTRAIN.RXDESKEW Local state diagram

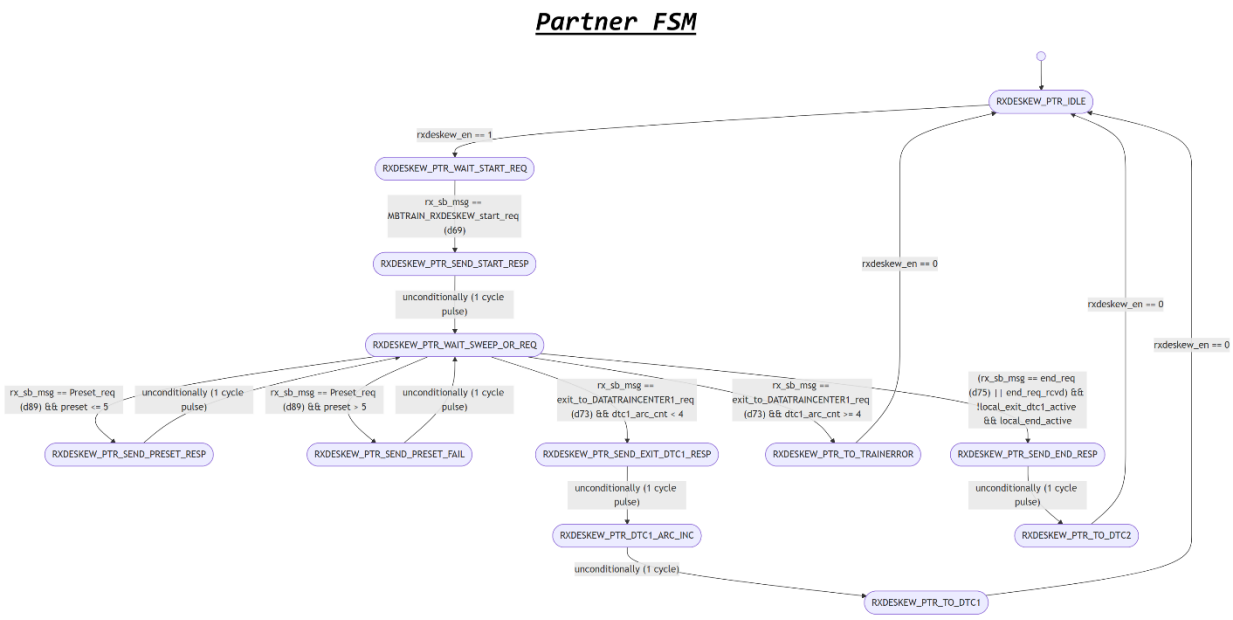


Figure 6-23: MBTRAIN.RXDESKEW Partner state diagram

DATATRAINCENTER2

DATATRAINCENTER2 performs a second pass of TX data PI centering after RXDESKEW has applied its equalization corrections. Having accounted for deskew-corrected lane timing, this pass refines the PI codes for all active data lanes. On completion dtc2_done is asserted and the controller advances to LINKSPEED.

Table 6-23: MBTRAIN.DATATRAINCENTER2 Interface Signals

Signal	Direction	Width	Description
mb_tx_clk_lane_sel	Output	2	01b if high-speed/continuous-clock, else 00b.
mb_tx_data_lane_sel / mb_tx_val_lane_sel / mb_tx_trk_lane_sel	Output	2 each	Held low; sweep engine active.
mb_rx_clk_lane_sel / mb_rx_data_lane_sel / mb_rx_val_lane_sel	Output	1 each	Enabled.
mb_rx_trk_lane_sel	Output	1	Disabled.
phy_tx_data_pi_phase_ctrl	Output	Enum	Transmit PI centering for Data lanes, Pass 2.
dtc2_start_req / resp, end_req / resp	Output/Input	Enum	Frames the second centering pass; transitions to LINKSPEED.

LINKSPEED

LINKSPEED verifies link stability at the current operating speed by running the D2C point test and evaluating per-lane pass/fail results against the established thresholds. The per-lane pass vector is latched into linkspeed_success_lanes for consumption by the subsequent REPAIR substate if width degradation is required. Based on the evaluation, LINKSPEED asserts one of four routing outputs: linkspeed_linkinit_req (stable, proceed to LINKINIT), linkspeed_speedidle_req (unstable, degrade speed), linkspeed_repair_req (unstable at minimum speed, degrade width), or linkspeed_phyretrain_req (parameter mismatch detected, initiate PHYRETRAIN). The controller latches these requests and takes the highest-priority exit.

Table 6-24: MBTRAIN.LINKSPEED Interface Signals

Signal	Direction	Width	Description
mb_tx_clk_lane_sel	Output	2	Speed-conditional normally; forced Electrical Idle (11b) if lcl_tx_elec_idle is asserted after a local error request.
mb_tx_data_lane_sel / mb_tx_val_lane_sel / mb_tx_trk_lane_sel	Output	2 each	Held low normally; Electrical Idle if lcl_tx_elec_idle asserted.
mb_rx_clk_lane_sel / mb_rx_data_lane_sel / mb_rx_val_lane_sel	Output	1 each	Enabled normally; forced disabled if ptr_rx_elec_idle asserted.
mb_rx_trk_lane_sel	Output	1	Disabled in all cases.
active_rx_lanes / width_degrade_feasible	Input	16/1	Determines whether a width-degrade path to REPAIR is feasible.
PHY_IN_RETRAIN	Input	1	Set when the link entered MBTRAIN via the PHY-retrain path.
PHY_IN_RETRAIN_rst	Output	1	Clears the flag after the error-response handshake.
params_changed	Input	1	If asserted with PHY_IN_RETRAIN, LINKSPEED returns to PHYRETRAIN instead of LINKINIT.
speedidle_req / repair_req / linkinit_req / phyretrain_req	Output	1 each	Resolves the stability outcome to the next state/substate.
linkspeed_error_req	Output	Enum	{error req}: sent on link-stability failure.
linkspeed_error_resp	Input	Enum	{error resp}: drives the lcl_tx_elec_idle / ptr_rx_elec_idle posture.

REPAIR

REPAIR negotiates and applies width degradation when LINKSPEED determines that lane failures cannot be resolved by a speed change. It uses the unit_negotiated_lanes module, instantiated inside wrapper_REPAIR, to evaluate the per-lane pass vector from LINKSPEED and determine which contiguous half or quarter of the active lane set is error-free. The local die computes its preferred degraded lane map code and exchanges it with the remote die over sideband. The agreed lane map (the intersection of local and partner codes) is applied as the new operational lane configuration for both TX and RX, driving the active_rx_lanes output used by subsequent MBTRAIN sessions. If no feasible degraded lane map exists (both local and partner codes are 3'b000), REPAIR asserts trainerror_req and the controller routes to TRAINERROR.

Table 6-25: MBTRAIN.REPAIR Interface Signals

Signal	Direction	Width	Description
mb_tx_clk_lane_sel	Output	2	00b; held differential/simultaneous low per specification.
mb_tx_data_lane_sel / mb_tx_val_lane_sel / mb_tx_trk_lane_sel	Output	2 each	Held low.
mb_rx_clk_lane_sel	Output	1	Enabled.
mb_rx_data_lane_sel / mb_rx_val_lane_sel / mb_rx_trk_lane_sel	Output	1 each	Disabled.
linkspeed_success_lanes	Input	16	Per-lane pass vector from the preceding LINKSPEED test.
repair_degrade_req	Output	3	Locally determined degrade-negotiation code (analogous to MBINIT.REPAIRMB).
repair_degrade_resp	Input	3	Partner degrade-negotiation code; 3'b000 from either side exits to TRAINERROR.
mb_width_deg_tx / mb_width_deg_rx	Output	3/3	Applies the agreed degraded lane map; carried into the TXSELFCAL re-entry.

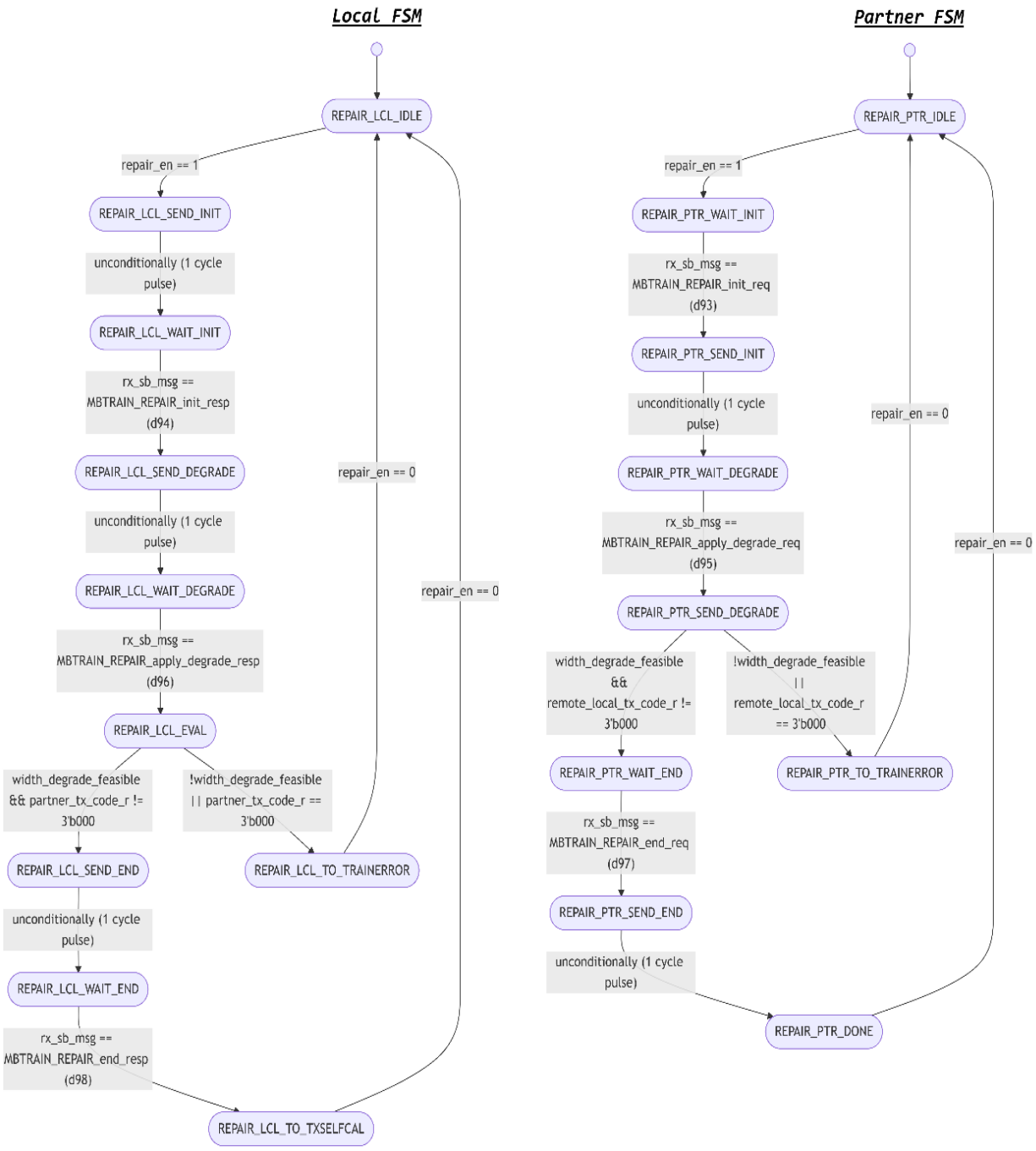


Figure 6-24: MBTRAIN.REPAIR state diagram

6.2.7 LINKINIT

LINKINIT is the final initialization handshake before normal data traffic begins. It exchanges the adapter and link management sideband messages required to complete protocol-layer initialization, verifies that both dies are ready to enter ACTIVE, and on successful completion asserts linkinit_done. A failure or timeout triggers the TRAINERROR entry handshake. LINKINIT does not perform any mainband pattern transmission; the mainband lanes remain in their post-MBTRAIN configuration throughout this state.

Table 6-26: LINKINIT Interface Signals

Signal	Direction	Width	Description
linkinit_msg_send	Output	Enum	D2D Adapter/Link management messages required to establish RDI Active, sent to Partner.
linkinit_msg_recv	Input	Enum	Corresponding messages received from Partner.
pl_inband_pres	Output	1	Asserted while rdi_state_sts is Reset and LTSM status is LINKINIT.
reg_Link_Width_enable_status / reg_Link_Speed_enable_status	Output	4/4	Final negotiated width/speed captured as LINKINIT completes.
reg_PMO_enable_status / reg_SFES_enable_status	Output	1/1	Final agreed feature-enable status recorded for software visibility.
pl_state_sts	Output	4	Remains Reset until the RDI state machine transitions on ACTIVE entry.

6.2.8 ACTIVE

The ACTIVE sub-block enables normal protocol traffic on the mainband by asserting the mapper enable to the datapath (through ltsm_mapper_en, later gated by the RDI stall level in MainSM glue). It monitors four exit conditions: detection of a PHY framing error on the valid lane (o_valid_frame_error → pl_error → active_error → PHYRETRAIN), a retrain request from the Adapter or remote die (PHYRETRAIN), a power-management L1 entry request from the Adapter (L1), a power-management L2 entry request (L2), and any fatal error (TRAINERROR). The resolved next state is driven into the active_next_ltsm_state output, which the unit_ltsm_controller consumes to select the correct exit from CTRL_ACTIVE.

Table 6-27: ACTIVE Interface Signals

Signal	Direction	Width	Description
pl_state_sts	Output	4	0001b Active; 0011b Active.PMNAK if a PM request is declined.
lp_state_req	Input	4	Active, L1, L2, LinkReset, Retrain, Disabled, NOP encodings.
pl_inband_pres	Output	1	Remains asserted throughout Active.
pl_clk_req	Output	1	Clock-gating request usable during idle intervals.
lp_clk_ack	Input	1	Clock-gating acknowledgment.
lp_wake_req	Input	1	Wake-up handshake preceding an Adapter-initiated state change.
pl_wake_ack	Output	1	Wake-up acknowledgment.
pl_stallreq	Output	1	Flit-flush request before a graceful exit from Active.
lp_stallack	Input	1	Flit-flush acknowledgment (STALLACK).
pl_error	Output	1	Receive-path framing error; triggers Retrain when detected.
mb_data_path	Input/Output	N×8	Mission-mode Flit data transfer at the negotiated width/speed.
pm_req	Output	Enum	{LinkMgmt.RDI.Req.PM}: power-management request sent to Partner.
pm_resp / pm_nak	Input	Enum	{LinkMgmt.RDI.Rsp.PM} / {LinkMgmt.RDI.Rsp.PMNAK}: response governing L1/L2 entry or rejection.

6.2.9 L1_L2

The L1 sub-block manages the shallow power management state, in which the mainband clock is gated but negotiated link parameters are preserved. On L1 entry the RDI_SM asserts the gating enable through lclk_g, which suppresses gated_lclk and eliminates dynamic power in all mainband parallel-domain logic. L1 exit is triggered by the Adapter through lp_state_req and leads the LTSM controller to CTRL_MBTRAIN with mbtrain_speedidle_req latched, so the training re-entry begins at SPEEDIDLE. The L2 sub-block manages the deeper power management state. On L2 exit the full link initialization sequence must be repeated; the controller therefore transitions to CTRL_RESET, where the 4 ms dwell timer and the software start trigger must both be satisfied before training begins again.

Table 6-28: L1 Interface Signals

Signal	Direction	Width	Description
pl_state_sts	Output	4	0100b L1.
lp_state_req	Input	4	L1 request/exit encodings.
pl_inband_pres	Output	1	Remains asserted through L1.
pm_entry_req	Output	Enum	{LinkMgmt.RDI.Req.PM}: PM entry handshake preserving negotiated link parameters.
pm_entry_resp	Input	Enum	{LinkMgmt.RDI.Rsp.PM}: partner acknowledgment.
pl_clk_req	Output	1	Clock-gating handshake associated with the shallow power state.
lp_clk_ack	Input	1	Clock-gating acknowledgment.
speedidle_req (L1 exit)	Output	1	On exit, LTSM transitions directly to MBTRAIN and re-enters at SPEEDIDLE.

Table 6-29: L2 Interface Signals

Signal	Direction	Width	Description
pl_state_sts	Output	4	1000b L2.
lp_state_req	Input	4	L2 request/exit encodings.
pl_inband_pres	Output	1	Remains asserted through L2.
pm_entry_req	Output	Enum	{LinkMgmt.RDI.Req.PM}: PM entry handshake for the deep power state.
pm_entry_resp	Input	Enum	{LinkMgmt.RDI.Rsp.PM}: partner acknowledgment.
l2_exit_to_reset	Output	1	L2 exit leads to RESET; full link initialization sequence is repeated.
reg_L2SPD_support_local_cap / ctrl	Input	1/1	L2 Sideband Power Down capability/control bits negotiated per protocol; sideband power-down mechanism itself is excluded from this design.

6.2.10 PHYRETRAIN

PHYRETRAIN is entered from ACTIVE when a retrain condition is detected. The sub-block exchanges the {PHYRETRAIN} sideband messages with the remote die to negotiate the re-entry point for the subsequent MBTRAIN visit. Three re-entry points are defined in the RTL: TXSELFCAL (when no speed change and no width change are needed), SPEEDIDLE (when a speed change is required), and REPAIR (when width degradation is required). The negotiated re-entry point drives the corresponding mbtrain_txselfcal_req, mbtrain_speedidle_req, or mbtrain_repair_req input that is sampled by unit_MBTRAIN_ctrl when mbtrain_en rises. On completion, phyretrain_done leads the controller back to CTRL_MBTRAIN. A timeout or error during PHYRETRAIN triggers the TRAINERROR entry handshake.

Table 6-30: PHYRETRAIN Interface Signals

Signal	Direction	Width	Description
pl_state_sts	Output	4	1011b Retrain.
retrain_source	Input	Enum	Retrain-condition source: Adapter request, PHY framing error, remote-die request, or MBTRAIN.LINKSPEED link-quality determination.
retrain_nego_req	Output	Enum	Retrain-negotiation message; carries locally resolved re-entry encoding (REPAIR > SPEEDIDLE > TXSELFCAL priority).
retrain_nego_resp	Input	Enum	Partner's resolved re-entry encoding.
mbtrain_txselfcal_req / speedidle_req / repair_req	Output	1 each	Drives the resolved MBTRAIN re-entry point once negotiation completes.
PHY_IN_RETRAIN	Output	1	Set on entry so a subsequent LINKSPEED parameter-change detection routes back to PHYRETRAIN.
tx_adjustment_runtime_recal	—	—	Defined by specification for PHYRETRAIN; not implemented (excluded feature).

6.2.11 D2C Shared Sweep Engine

The shared Die-to-Continuity sweep engine (wrapper_D2C_PT / wrapper_D2C_PT_top) is instantiated at the LTSM wrapper level and shared across all substates that require a pattern sweep. It manages both the local-die TX-initiated point test and the partner-die RX-received point test, with the active role determined by which substate is driving the enable and configuration. The sweep engine outputs the swept code, best code, done signal, and minimum eye width back to the active substate through the shared sweep bus. It also outputs det_pat_rcvd, d2c_perlane_pass, d2c_aggr_pass, and d2c_val_pass, which substates use to evaluate lane quality. Only one substate drives the local_sweep_en or partner_sweep_en inputs at any time, enforced by the single-active-enable discipline of unit_MBTRAIN_ctrl.

6.2.12 TRAINERROR & TRAINERROR Handshake

TRAINERROR may be entered from any state except RESET. The entry mechanism differs by origin state. From SBINIT and PHYRETRAIN, the transition is direct upon fault detection or timeout, since the sideband channel may not yet be reliable enough for a coordinated handshake. From MBINIT, MBTRAIN, and LINKINIT, the entry requires a two-way sideband handshake defined by the UCIe specification (Section 4.5.3.8), implemented in trainerror_handshake.sv. This module handles both the initiator role (local error triggers the handshake) and the responder role (remote error received first), with collision-safe arbitration when both dies detect an error simultaneously. The handshake sends a {TRAINERROR} request sideband message and waits for the corresponding response before asserting te_handshake_done to the controller. Once inside TRAINERROR, the sub-block performs its own sideband exchange and always exits to RESET via trainerror_done.

Table 6-31: TRAINERROR Interface Signals

Signal	Direction	Width	Description
mb_tx_clk/data/val/trk_lane_sel	Output	2 each	11b Electrical Idle on all mainband transmitters.
sb_active	Output	1	Sideband channel remains active to permit the entry handshake to complete.
sbinit_error / mbinit_error / mbtrain_error / linkinit_error / active_error	Input	1 each	Per-origin error indications; SBINIT/PHYRETRAIN transition directly, others require the handshake below.
te_handshake_req	Output	Enum	Section 4.5.3.8 two-way sideband error-acknowledgment handshake, sent to Partner.
te_handshake_resp / te_handshake_done	Input	Enum/1	Partner acknowledgment and completion gating the move to TRAINERROR.
pl_trainerror	Output	1	Asserted to indicate a training error has been detected.
pl_state_sts	Output	4	1010b LinkError.
reg_state_n .. state_n_minus_3	Output	8 each	Records the LTSM state history at the point of fault (Error Log 0/1).
timeout_8ms_occured / second_timeout_occured	Input	1/1	First timeout drives direct/handshake-gated entry; second timeout forces direct entry.

6.3 RDI SM Design

6.3.1 RDI SM Architecture

The RDI SM implements the PHY side of the Raw Die-to-Die Interface state protocol. It runs the RDI power and link state sequencing, executes the clock, wake, stall, and active entry handshakes with the D2D Adapter, gates the mainband clock for power management, and exchanges link-management messages over the sideband channel. The RDI_SM is structured as five cooperating sub-blocks: wrapper_sm (state sequencing), wrapper_handshake_logic (protocol handshakes), unit_gating_logic (clock gate control), unit_signal_transition_detector (output update sequencing), and the message handler group (unit_msg_handler, message_timeout_handler, unit_message_send_MUX). All blocks operate in the lclk domain.

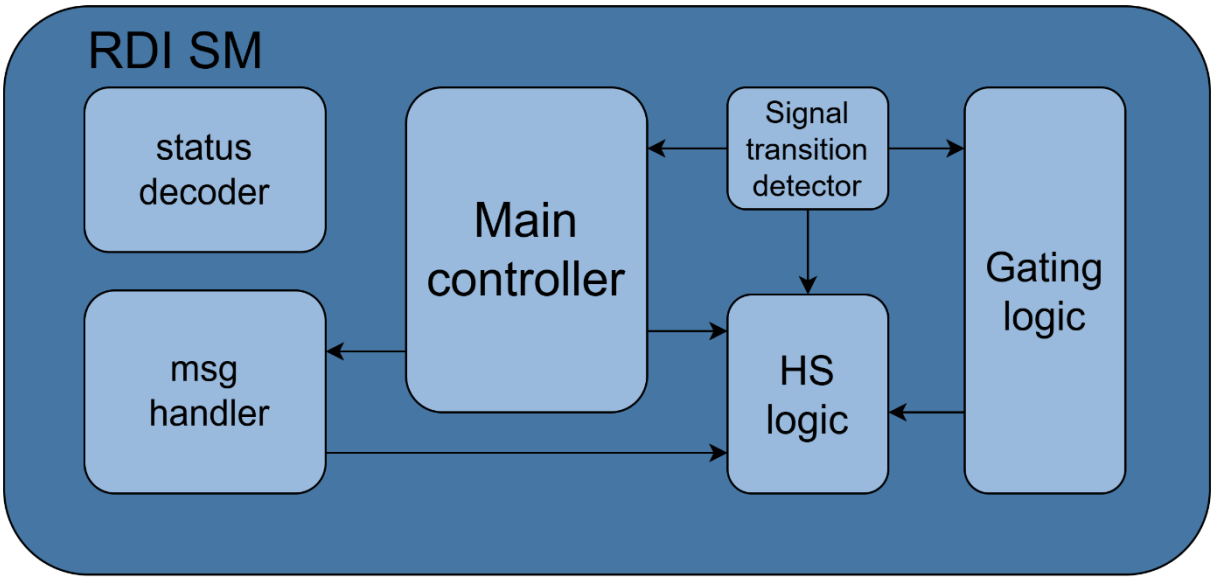


Figure 6-25: RDI SM Architecture

6.3.2 State Machine

SM is the core RDI state sequencer. It instantiates nine per-state unit modules (unit_reset_state, unit_active_state, unit_active_pmnak_state, unit_retrain_state, unit_L1_state, unit_L2_state, unit_linkreset_state, unit_linkerror_state, unit_disabled_state), a shared unit_Timer that provides 1 µs and 16 ms timing references, and a message_timeout_handler that monitors the 8 ms SB message handshake timeout. Each state module receives the current rdi_state_sts and the relevant input signals, evaluates its exit conditions, and drives a next_state output. An internal main controller (unit_main_controller) arbitrates among the state next_state outputs to produce the registered rdi_state_sts, which is exposed as the rdi_state output to the LTSM. The wrapper also computes the aggregated stall_req as the logical OR of Active_stall_req and Active_PMNAK_stall_req, since both the Active and Active_PMNAK states may independently request a stall.

The message_timeout_handler monitors the sideband link-management message exchange timer. When a state initiates a message handshake and the partner response is not received within the allowed 8 ms window, the handler asserts sb_msg_timeout, which causes the state to transition to LinkError. A le_mux_sel signal selects between the per-state message_send outputs and the handler's le_message_send, ensuring that the error message is transmitted before the state changes.

Table 6-32: State Machine Interface signals

Signal	Direction	Width	Description
lclk	Input	1	RDI domain clock
rst_n	Input	1	Active-low reset
state_sts	Input	Enum	Current LTSM top-level state (LTSM_state_e); used by per-state modules to condition transitions
pl_error	Input	1	Valid frame error from mainband (mb_valid_frame_error); triggers Active → error path
lp_linkerror	Input	1	Adapter link-error indication; causes LinkError entry
lp_state_req	Input	Enum	Adapter-requested RDI state; consumed by per-state modules
message_receive	Input	Enum	Decoded received RDI link-management message from unit_msg_handler
Active_handshake_done	Input	1	Active-entry handshake completed (from wrapper_handshake_logic)
stall_done	Input	1	Stall handshake STALLACK reached (from unit_stall_handshake)
sb_msg_timeout	Input	1	8 ms SB message timeout fired (from message_timeout_handler)
stall_req	Output	1	Aggregated stall request (Active_stall_req | Active_PMNAK_stall_req)
Active_handshake_strt	Output	1	Start active-entry handshake (OR of per-state handshake triggers)
message_send	Output	Enum	Selected RDI link-management message to transmit (via unit_message_send_MUX)
trainerror	Output	1	PHY training error level; forwarded to unit_signal_transition_detector
phyinrecenter	Output	1	PHY-in-recentering level; forwarded to unit_signal_transition_detector
pm_exit	Output	1	Power-management exit flag; forwarded to wrapper_handshake_logic
inband_pres	Output	1	In-band presence indicator; forwarded to unit_signal_transition_detector and gating_logic

Signal	Direction	Width	Description
rdi_state_sts	Output	Enum	Internal registered RDI state; forwarded as rdi_state to LTSM and as pl_state_sts input to unit_signal_transition_detector

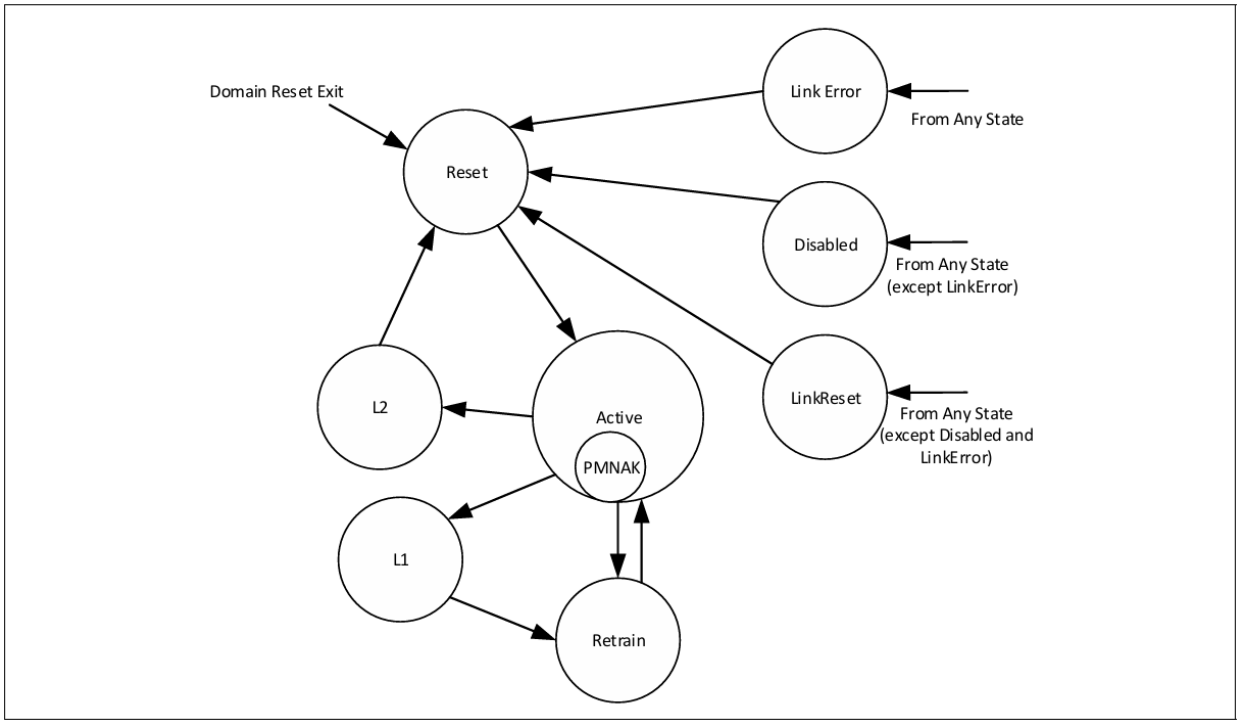


Figure 6-26: RDI State Machine

6.3.3 Handshake logic

Handshake_logic groups the four protocol handshakes that the RDI_SM must execute with the Adapter.

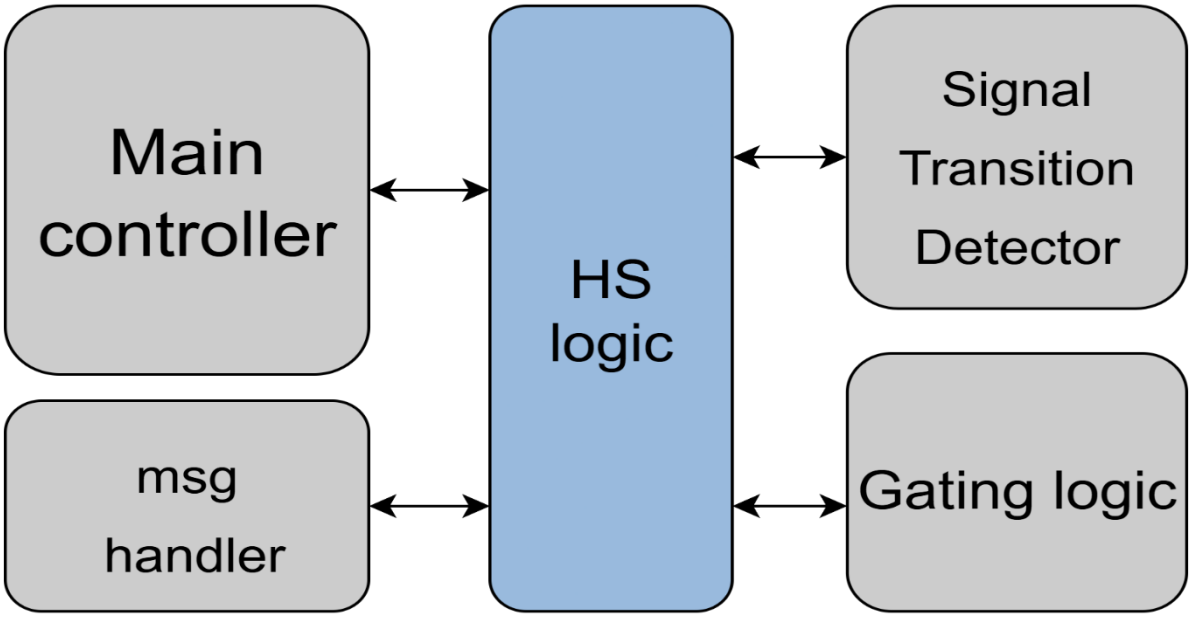


Figure 6-27: Handshake Logic Architecture

The AWAKE handshake responds to an Adapter wake request (lp_wake_req) by waiting for the gating logic to confirm that the mainband clock has been restored (ungating_done), then asserting pl_wake_ack. The STALL handshake is initiated internally when the state machine requires a data-path stall: it asserts pl_stallreq, waits for lp_stallack, then asserts stall_done. The CLK handshake sequences the sideband traffic window: it starts when clk_handshake_strt is asserted (triggered internally when a change in pl_state_sts, pl_phyinrecenter, pl_inband_pres, or pl_trainerror is pending), drives pl_clk_req, and waits for lp_clk_ack before asserting clk_handshake_done. The ACTIVE ENTRY handshake coordinates the transition into the Active state, asserting Active_message_send and waiting for Active_handshake_done before completing.

Table 6-33: Handshake Logic Interface signals

Signal	Direction	Width	Description
lclk	Input	1	RDI domain clock
rst_n	Input	1	Active-low reset
lp_wake_req	Input	1	Adapter wake request (async to lclk); triggers AWAKE handshake
ungating_done	Input	1	Clock fully ungated (from unit_gating_logic); permits pl_wake_ack
pl_wake_ack	Output	1	PHY wake acknowledgment to adapter
ungating_req	Output	1	Request to unit_gating_logic to begin clock ungate sequence
stall_req	Input	1	Stall request from wrapper_sm (Active or Active_PMNAK state)
lp_stallack	Input	1	Adapter stall acknowledgment
mapper_en	Input	1	Mapper enable level; cleared after stall to reset stall_done_latched
pl_stallreq	Output	1	PHY stall request to adapter (via unit_stall_handshake)
stall_done	Output	1	Stall handshake reached STALLACK
stall_done_latched	Output	1	Mapper-stall level: held until mapper_en falls
Active_handshake_strt	Input	1	Start active-entry handshake (from wrapper_sm)
message_receive	Input	Enum	Received RDI message; used to detect RDI_ACTIVE_RSP
pm_exit	Input	1	Power-management exit flag; qualifies active-entry path from L1/L2
inband_pres	Input	1	In-band presence; determines whether active-entry sends RDI_ACTIVE_REQ
Active_handshake_done	Output	1	Active-entry handshake complete
Active_message_send	Output	Enum	Message to send during active-entry (RDI_ACTIVE_REQ or NOP)
clk_handshake_strt	Input	1	Start CLK handshake (signal_transition || traffic_req, from RDI_SM top)
lp_clk_ack	Input	1	Adapter clock acknowledgment
pl_clk_req	Output	1	PHY clock request to adapter
clk_handshake_done	Output	1	CLK handshake complete; triggers unit_signal_transition_detector output update

awake Handshak

unit_awak_handshake implements the AWAKE handshake, which the RDI_SM uses to restore the mainband clock after a power-management-gated period before acknowledging the adapter's wake request. The module is a three-state Moore FSM: IDLE, UNGATING, and ACK.

In IDLE, the module monitors lp_wake_req from the adapter. When the adapter asserts lp_wake_req to request link wake-up, the FSM transitions to UNGATING and asserts ungating_req. The ungating_req output notifies unit_gating_logic to begin restoring the mainband gated clock, which requires driving lclk_g high and waiting for the clock to stabilize. Once unit_gating_logic confirms that the clock is fully available by asserting ungating_done, the FSM transitions to ACK and asserts pl_wake_ack to the adapter, completing the wake handshake. The FSM remains in ACK while lp_wake_req is held high, then returns to IDLE when the adapter deasserts lp_wake_req, closing the handshake window.

Table 6-34: awake Handshake Interface Signals

Signal	Direction	Width	Description
lclk	Input	1	RDI domain clock
rst_n	Input	1	Active-low reset; returns FSM to IDLE
lp_wake_req	Input	1	Adapter wake request; rising edge triggers transition from IDLE to UNGATING
ungating_done	Input	1	Clock fully restored (from unit_gating_logic); triggers transition from UNGATING to ACK
ungating_req	Output	1	Asserted in UNGATING state; requests unit_gating_logic to restore the mainband clock
pl_wake_ack	Output	1	Asserted in ACK state; informs the adapter that the PHY has restored the clock and acknowledged the wake request

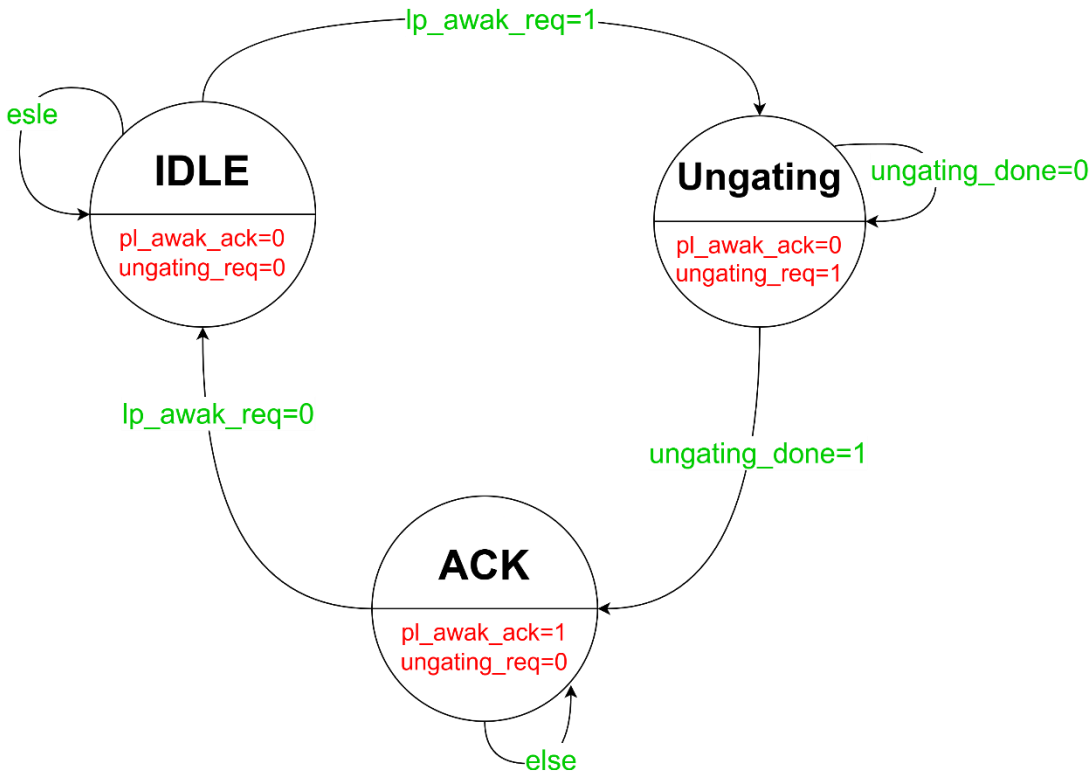


Figure 6-28: AWAKE Handshake State Diagram

CLK Handshake

unit_clk_handshake implements the CLK handshake, which is required by the UCIe RDI protocol before the PHY may update the adapter-visible output signals pl_state_sts, pl_phyinrecenter, pl_inband_pres, or pl_trainerror. The handshake ensures the adapter has acknowledged a pending output change before those signals are allowed to transition. The module is a three-state Moore FSM: IDLE, REQ, and DONE.

In IDLE, the module waits for clk_handshake_strt to be asserted by the RDI_SM top-level logic. clk_handshake_strt is driven by the combinational expression signal_transition || traffic_req, where signal_transition is produced by unit_signal_transition_detector when any tracked output has changed from its last registered value, and traffic_req is asserted by SideBand_Top when upstream sideband traffic is pending. On clk_handshake_strt, the FSM transitions to REQ and asserts pl_clk_req to the adapter. The FSM waits in REQ until the adapter responds with lp_clk_ack, at which point it transitions to DONE and asserts clk_handshake_done for one cycle. unit_signal_transition_detector uses the clk_handshake_done pulse to latch and present the updated pl_ output values to the adapter, completing the protected update sequence. The DONE

state returns to IDLE on the deassertion of clk_handshake_strt, ensuring the protocol window is closed before the next handshake can begin.

Table 6-35: clk_handshake Interface Signals

Signal	Direction	Width	Description
lclk	Input	1	RDI domain clock
rst_n	Input	1	Active-low reset; returns FSM to IDLE
clk_handshake_strt	Input	1	Start CLK handshake; driven by RDI_SM top as signal_transition || traffic_req. Held high during the full pending-output-update period
lp_clk_ack	Input	1	Adapter clock acknowledgment; advances FSM from REQ to DONE
pl_clk_req	Output	1	Asserted in REQ state; PHY clock request to adapter
clk_handshake_done	Output	1	Asserted in DONE state; triggers unit_signal_transition_detector to present latched pl_ output updates to the adapter

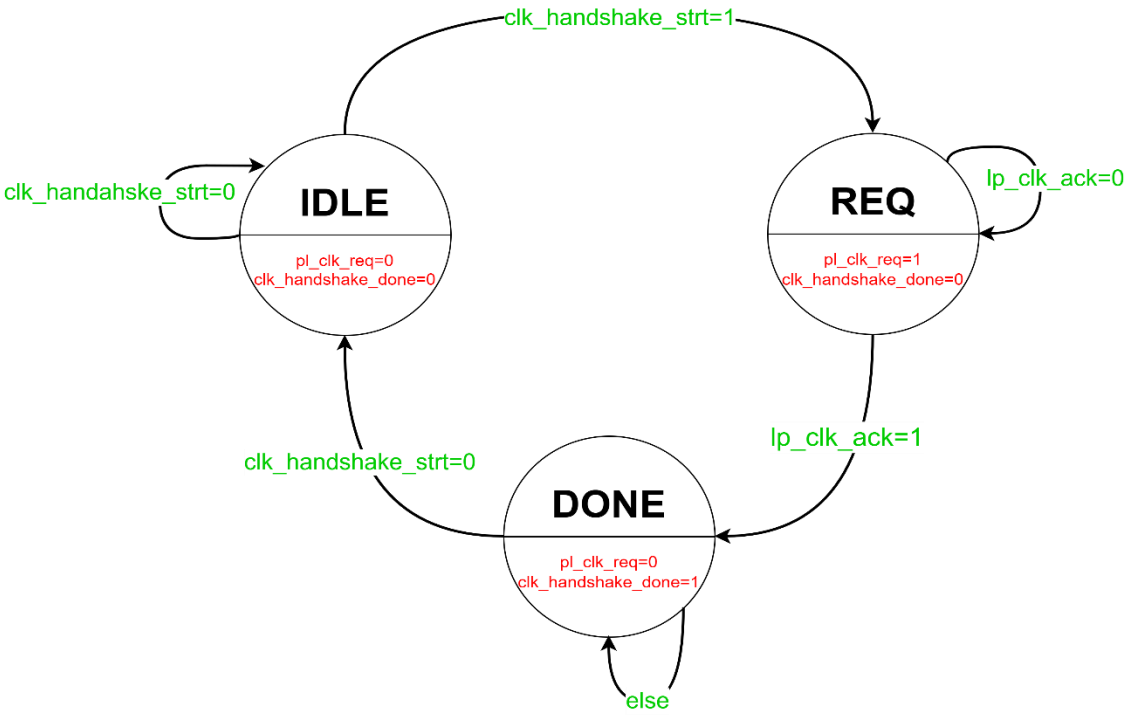


Figure 6-29: CLK Handshake State Diagram

Stall handshake

unit_stall_handshake implements the STALL handshake, through which the RDI_SM requests the adapter to pause mainband data flow before the PHY performs an operation — such as retraining

or a power-management transition — that requires the datapath to be quiescent. The module is a four-state Moore FSM: IDLE, STALLREQ, STALLACK, and STALLDONE.

In IDLE, the FSM waits for stall_req to be asserted from wrapper_sm. stall_req is the logical OR of Active_stall_req and Active_PMNAK_stall_req, since both the Active and Active_PMNAK states may independently require a stall. When stall_req is asserted, the FSM transitions to STALLREQ and asserts pl_stallreq to the adapter. The FSM waits in STALLREQ until the adapter confirms that it has ceased mainband traffic by asserting lp_stallack, at which point the FSM transitions to STALLACK. In STALLACK both pl_stallreq and stall_done are asserted: stall_done notifies wrapper_sm that the stall is confirmed and the datapath is safe to modify. The FSM exits STALLACK when stall_req is deasserted (the per-state logic has completed its operation and released the stall request), transitioning to STALLDONE. In STALLDONE, pl_stallreq remains deasserted and the FSM waits for lp_stallack to be released by the adapter, then returns to IDLE.

In addition to the FSM outputs, the module maintains the stall_done_latched register. This register is set when stall_done is asserted (in STALLACK) and cleared when mapper_en falls to zero. It is forwarded to MainSM glue logic, where it is ANDed with ltsm_mapper_en to produce the final mb_mapper_en signal. This level-hold behavior ensures that the mapper is held off for the full duration from stall acknowledgment until the LTSM itself lowers its own enable, without requiring the LTSM or wrapper_sm to directly track stall state.

Table 6-36: stall handshake Interface Signals

Signal	Direction	Width	Description
lclk	Input	1	RDI domain clock
rst_n	Input	1	Active-low reset; returns FSM to IDLE and clears stall_done_latched
stall_req	Input	1	Aggregated stall request (Active_stall_req | Active_PMNAK_stall_req); rising edge triggers transition from IDLE to STALLREQ
lp_stallack	Input	1	Adapter stall acknowledgment; advances FSM from STALLREQ to STALLACK. Also gates STALLDONE→IDLE exit: FSM waits until lp_stallack deasserts
mapper_en	Input	1	Mapper enable from MainSM; falling edge clears stall_done_latched, releasing the mapper-enable gate
pl_stallreq	Output	1	PHY stall request to adapter; asserted combinationally in both STALLREQ and STALLACK states
stall_done	Output	1	Asserted combinationally in STALLACK state; informs wrapper_sm that the adapter has acknowledged the stall and the datapath is quiescent
stall_done_latched	Output	1	Level register: set when stall_done is asserted, cleared when mapper_en falls. Forwarded to MainSM glue for mapper-enable gating

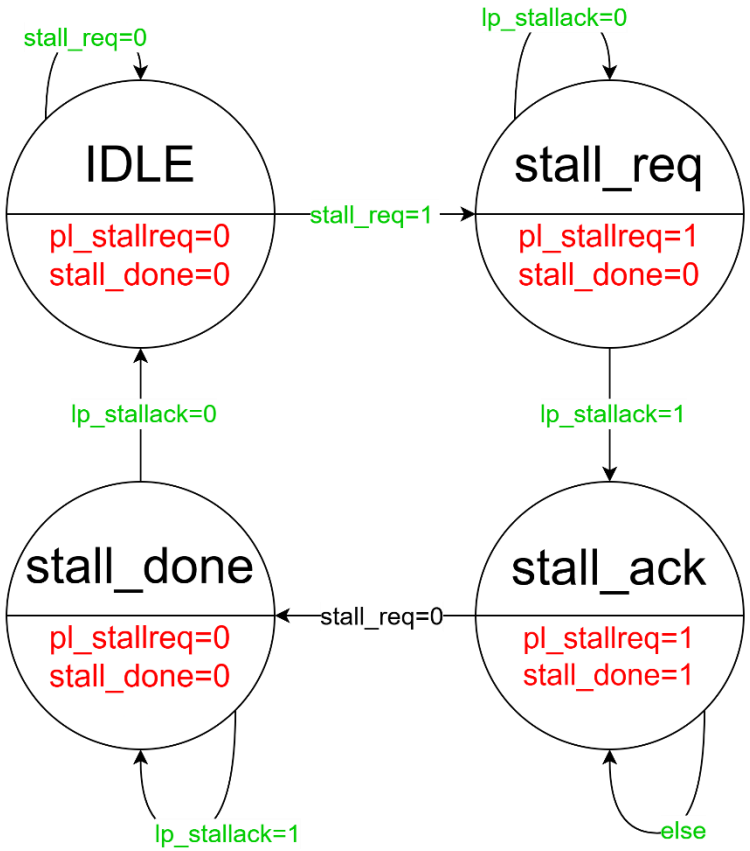


Figure 6-30: Stall Handshake State Diagram

active_handshake

unit_active_handshake implements the Active Entry handshake, which both PHY dies must complete before either may assert pl_state_sts = Active. The handshake requires both dies to exchange RDI_ACTIVE_REQ and RDI_ACTIVE_RSP sideband messages; the exact message ordering depends on which die initiates first and whether a collision occurs. The module resolves these ordering variations through a five-state FSM (IDLE, SEND_REQ, CHECK_MSG, SEND_RESP, DONE) augmented by a four-value flow register (FLOW_NONE, FLOW_0, FLOW_1, FLOW_2) and two message-latch registers (req_r, rsp_r).

Both req_r and rsp_r are unconditionally latched on every clock cycle whenever message_receive equals RDI_ACTIVE_REQ or RDI_ACTIVE_RSP respectively, regardless of the current FSM state. This pre-buffering ensures that a peer message arriving before the local FSM is ready to process it is not dropped.

In IDLE, the FSM waits for active_handshake_strt. Two entry arcs are defined. If active_handshake_strt is asserted and either req_r is not set or pm_exit is asserted, the FSM transitions to SEND_REQ and drives RDI_ACTIVE_REQ on active_message_send. The pm_exit condition overrides a pending req_r to allow the REQ to be transmitted even when a peer REQ has

already been received, which is needed during power-management exit sequences where both sides initiate simultaneously. If active_handshake_strt is asserted and req_r is already set (peer REQ arrived before the local FSM started) and pm_exit is not asserted, the FSM transitions directly to SEND_RESP, drives RDI_ACTIVE_RSP, and sets flow to FLOW_1, recording that the local response was sent before the local request.

SEND_REQ deasserts active_message_send to NOP for one cycle and transitions unconditionally to CHECK_MSG, giving unit_msg_handler one cycle to consume the request before the FSM monitors for the response.

CHECK_MSG evaluates the message-latch state under four conditions evaluated in priority order. First, if rsp_r is set and flow is not FLOW_1, the peer response has arrived in the standard sequence: the FSM remains in CHECK_MSG, sets flow to FLOW_0, and clears rsp_r, waiting for the peer's subsequent REQ before sending the local RSP. Second, if req_r is set, inband_pres is asserted, and flow is FLOW_0, the standard full exchange is in progress: the FSM transitions to SEND_RESP, drives RDI_ACTIVE_RSP, and clears req_r. Third, if req_r is set, inband_pres is asserted, and flow is not FLOW_0, a collision or overlap is occurring: the FSM transitions to SEND_RESP, drives RDI_ACTIVE_RSP, sets flow to FLOW_2 (collision in progress), and clears req_r. Fourth, if rsp_r is set and flow is exactly FLOW_1, the peer response has arrived after the local RSP was already sent in FLOW_1: the FSM transitions directly to DONE and clears rsp_r.

SEND_RESP handles three exit conditions determined by the current flow. In FLOW_1 (local RSP was sent first), the FSM transitions back to SEND_REQ and drives RDI_ACTIVE_REQ, completing the reversed-order exchange. In FLOW_0 (standard order complete), the FSM transitions to DONE. In FLOW_2 (collision), the FSM transitions to DONE only if rsp_r is already set (the outstanding peer RSP has arrived), otherwise it waits in SEND_RESP until it does. DONE asserts active_handshake_done for one cycle, clears both latches and the flow register, and returns to IDLE.

Table 6-37: active handshake Interface Signals

Signal	Direction	Width	Description
lclk	Input	1	RDI domain clock
rst_n	Input	1	Active-low reset; returns FSM to IDLE, clears flow, req_r, rsp_r, and active_message_send
active_handshake_strt	Input	1	Start the Active Entry handshake; aggregated from wrapper_sm per-state triggers (Reset, Retrain, L1, L2 each have an independent strt signal ORed together)
message_receive	Input	msg_no_e	Received RDI message from unit_msg_handler; unconditionally sets req_r when == RDI_ACTIVE_REQ and rsp_r when == RDI_ACTIVE_RSP on every clock cycle
pm_exit	Input	1	Power-management exit flag from wrapper_sm; when asserted in IDLE allows the SEND_REQ branch even if req_r is already set, handling simultaneous L1/L2 exit initiation by both dies
inband_pres	Input	1	In-band presence level; must be asserted for the FSM to transmit RDI_ACTIVE_RSP in CHECK_MSG (gates RSP transmission to the condition that the link is present)
active_message_send	Output	msg_no_e	RDI_ACTIVE_REQ in SEND_REQ; RDI_ACTIVE_RSP in applicable SEND_RESP transitions; NOP otherwise. Forwarded to unit_msg_handler.Active_message_send
active_handshake_done	Output	1	Combinationally asserted when state == DONE; informs wrapper_sm that both dies have completed the RDI_ACTIVE message exchange and pl_state_sts = Active may be asserted

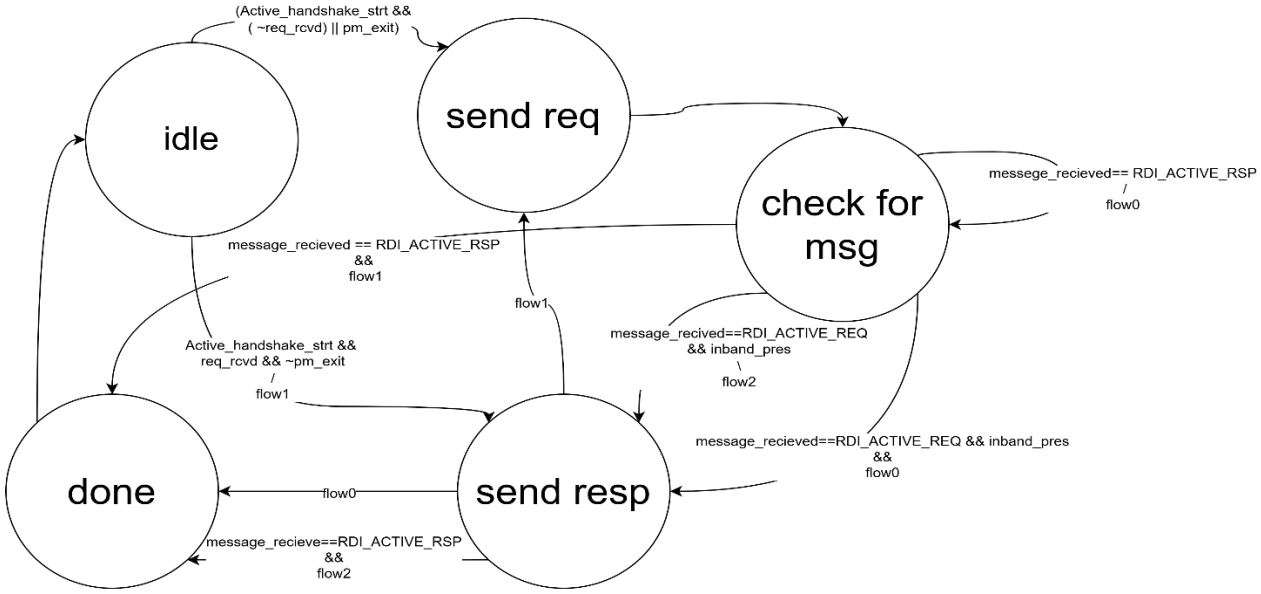


Figure 6-31: Active Handshake State Diagram

6.3.4 Gating Logic

unit_gating_logic drives lclk_g, the enable signal to the ICG cell that gates gated_lclk and thereby suppresses dynamic power in all mainband parallel-domain logic during power management states. The module implements UNGATING and GATING output phases. In the UNGATING phase (lclk_g = 1), the mainband clock is active. In the GATING phase (lclk_g = 0), the clock is suppressed. The gating condition is evaluated by checking the current pl_state_sts against the set of states in which gating is permitted (RESET, LinkReset, Disabled, and the L1/L2 power management states). The module additionally keeps lclk_g asserted through the RESET-to-SBINIT training start window, using phy_start_ucie_link_training_ctrl_out and sticky_sb_pattern_detected to determine when it is safe to allow gating in RESET. An ungating_req output is asserted when the gating logic determines that the clock should be restored, prompting the AWAKE handshake in wrapper_handshake_logic.

Table 6-38: Gating Logic Interface Siganls

Signal	Direction	Width	Description
lclk	Input	1	RDI domain clock
rst_n	Input	1	Active-low reset
inband_pres	Input	1	In-band presence indicator from wrapper_sm
phyinrecenter	Input	1	PHY-in-recentering flag from wrapper_sm
pl_clk_req	Input	1	Clock request (fed from clk handshake output)
ungating_req	Input	1	Ungating requested by handshake logic
phy_start	Input	1	Training start signal (from LTSM start control)
sticky_sb_pattern_detected	Input	1	Latched SBINIT pattern detection
pl_state_sts	Input	Enum	Current RDI state status (from signal_transition_detector)
lclk_g	Output	1	MB TX clock gate enable (1 = clock on, 0 = gated)
ungating_done	Output	1	Clock fully ungated; AWAKE handshake may complete

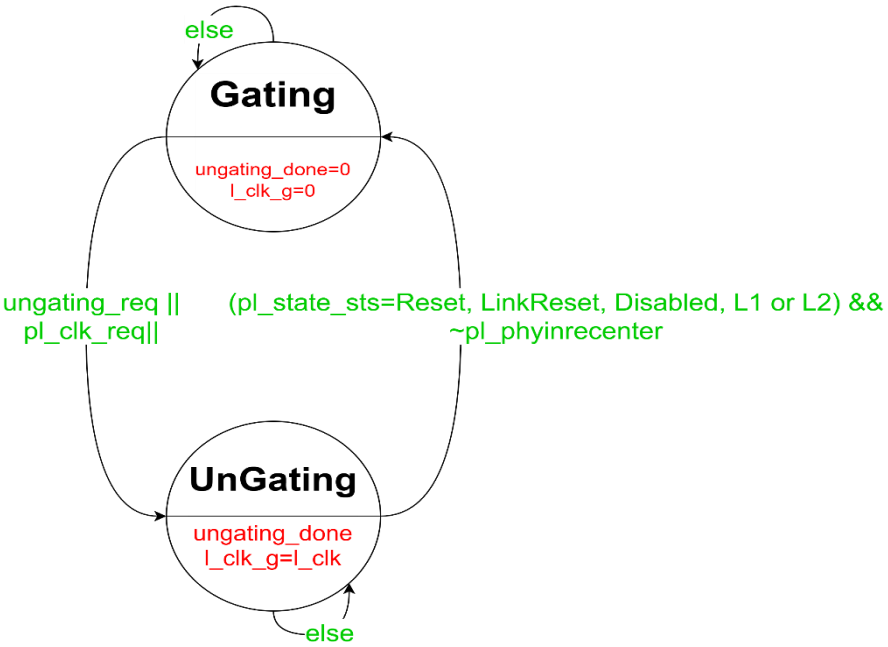


Figure 6-32: Gating Logic State Diagram

6.3.5 Signal Transition Detector

unit_signal_transition_detector manages the protocol requirement that updates to pl_state_sts, pl_phyinrecenter, pl_inband_pres, and pl_trainerror must be delivered to the Adapter through the clock handshake (pl_clk_req/lp_clk_ack) rather than changing asynchronously. The module

implements a two-state FSM: IDLE and CLK_HANDSHAKE. In IDLE, it continuously monitors whether any of the four tracked signals has changed from its last registered output value. On detecting a change, it transitions to CLK_HANDSHAKE and asserts the signal_transition flag, which triggers the CLK handshake in wrapper_handshake_logic. It remains in CLK_HANDSHAKE until clk_handshake_done is received, at which point it registers the updated values of all four signals onto the corresponding pl_ outputs and returns to IDLE. This mechanism ensures that the Adapter always observes consistent, edge-aligned updates rather than combinational glitches.

Table 6-39: Signal Transition Detector Interface signals

Signal	Direction	Width	Description
lclk	Input	1	RDI domain clock
rst_n	Input	1	Active-low reset
phyinrecenter	Input	1	Internal PHY-in-recentering level from wrapper_sm
inband_pres	Input	1	Internal in-band presence level from wrapper_sm
trainerror	Input	1	Internal training error level from wrapper_sm
clk_handshake_done	Input	1	Clock handshake complete; latch and update pl_ outputs
rdi_state_sts	Input	Enum	Internal RDI state from wrapper_sm
pl_phyinrecenter	Output	1	Registered PHY-in-recentering output to adapter
pl_inband_pres	Output	1	Registered in-band presence output to adapter
pl_trainerror	Output	1	Registered training error output to adapter
pl_state_sts	Output	Enum	Registered RDI state status output to adapter
signal_transition	Output	1	Any tracked signal has changed; CLK handshake is needed

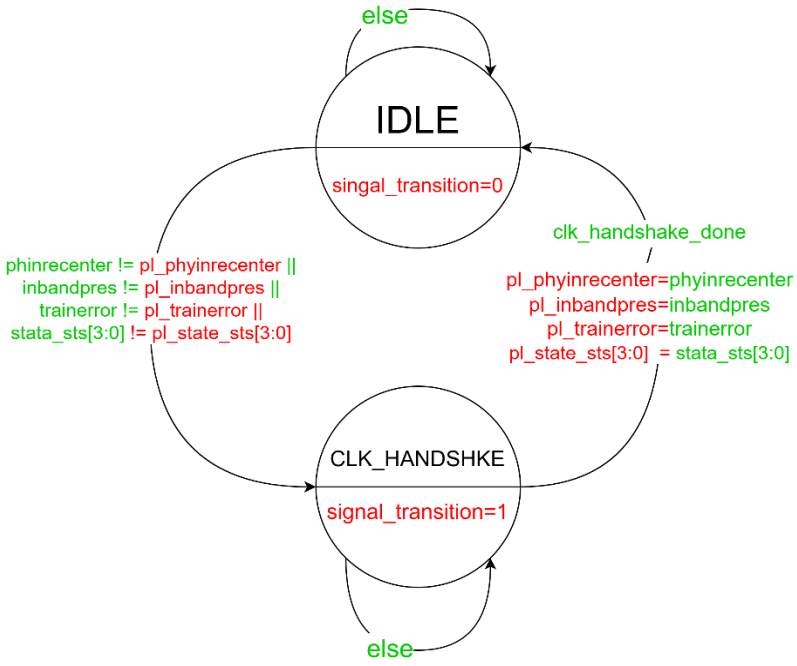


Figure 6-33: Signal Transition Detector State Diagram

6.3.6 Status Decoder

unit_status_decoder is a purely combinational block that derives the three adapter-facing link capability and status signals (pl_max_speedmode, pl_speedmode, pl_lnk_cfg) from register-file read-back values. It has no registered state and introduces no latency; its outputs update combinationally whenever its inputs change. The block sits between the Reg_File status outputs and the RDI_SM adapter interface, translating the raw four-bit register-file fields into the three-bit encoded formats expected on the RDI output boundary.

pl_max_speedmode is asserted when the maximum negotiated link speed exceeds 32 GT/s. The threshold is encoded in bits [2:0] of the four-bit UCIe Link Capability field: values 0h through 5h correspond to data rates up to 32 GT/s, while values 6h and 7h correspond to 48 GT/s and 64 GT/s respectively. The decoder therefore asserts pl_max_speedmode when bits [2:0] of the capability field are greater than 3'd5. pl_lnk_cfg carries the negotiated active lane-width configuration, taken directly from bits [2:0] of the UCIe Link Status field at bits [10:7]. pl_speedmode carries the current operating speed mode, taken from bits [2:0] of the UCIe Link Status field at bits [17:11]. Both pl_lnk_cfg and pl_speedmode are wired assignments with no additional decode logic.

Table 6-40: Status Decoder Interface signals

Signal	Direction	Width	Description
UCIe_Link_DVSEC_UCIe_Link_Capability_7_downto_4	Input	4	UCIe Link Capability register bits [7:4] from Reg_File (max link speed capability encoding)
UCIe_Link_DVSEC_UCIe_Link_Status_10_downto_7	Input	4	UCIe Link Status register bits [10:7] from Reg_File (negotiated link width status encoding)
UCIe_Link_DVSEC_UCIe_Link_Status_17_downto_11	Input	4	UCIe Link Status register bits [17:11] from Reg_File (negotiated link speed status encoding)
pl_max_speedmode	Output	1	1 when max negotiated speed > 32 GT/s (capability bits[2:0] > 3'd5); valid only during Reset-to-Active transition
pl_lnk_cfg	Output	3	Active link width configuration (Status[10:7][2:0]); valid when pl_state_sts is Active or Retrain
pl_speedmode	Output	3	Current link operating speed mode (Status[17:11][2:0]); valid when pl_state_sts is Active or Retrain

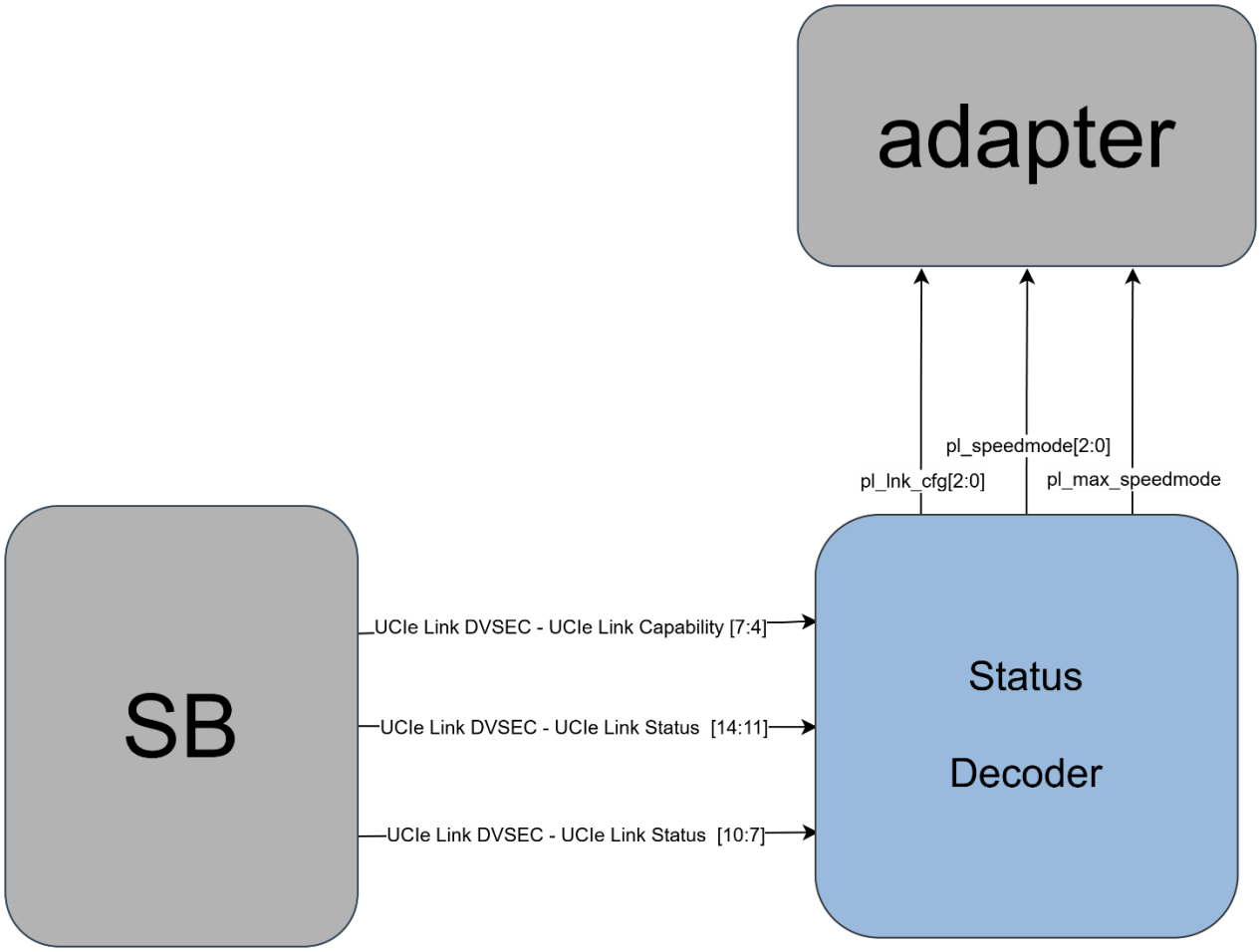


Figure 6-34: State Decoder diagram

6.3.7 Message handler

unit_msg_handler is a four-state FSM that serializes the outgoing and incoming RDI sideband link-management message paths so that only one message transaction is active at any given clock cycle. The four states are IDLE, LnkMsgS (Link Layer Message Send), LnkMsgR (Link Layer Message Receive), and ActvHsS (Active Handshake Send).

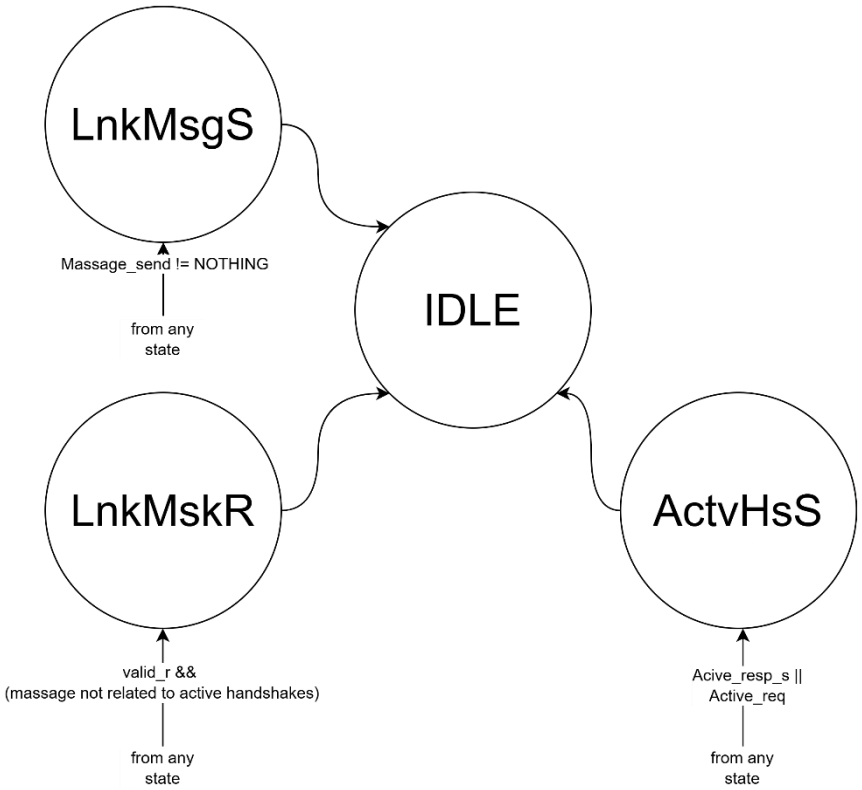


Figure 6-35: Message Handler State machine

A one-entry receive capture register (pending_rx_msg, pending_rx_valid) decouples the FSM from the arrival rate of incoming messages on the valid_r / Link_Mgmt_Msg_Received interface. Every assertion of valid_r unconditionally latches the received message number into pending_rx_msg and sets pending_rx_valid, ensuring that a message arriving while the FSM is processing a prior transaction is buffered rather than silently dropped. A single unified always_ff block governs the capture register to prevent multi-driver conflicts: consume (clear) takes priority over capture (set) on the same clock cycle, preventing a simultaneously arriving message from re-asserting a flag that the FSM is clearing in the same cycle.

State transitions follow a fixed three-way priority that is identical in every state. The highest priority is a non-NOP Message_send (driven by unit_message_send_MUX), which causes the FSM to enter or remain in LnkMsgS, assert valid_s, and drive Link_Mgmt_Msg_Send with the selected message. The second priority is a pending_rx_valid, which causes the FSM to enter LnkMsgR, present the buffered message on Message_receive, and clear pending_rx_valid. The lowest priority is a non-NOP Active_message_send (driven by wrapper_handshake_logic), which causes the FSM to enter

ActvHsS and assert either RDI_ACTIVE_REQ or RDI_ACTIVE_RSP on Link_Mgmt_Msg_Send depending on which encoding Active_message_send carries. If none of the three conditions is met, the FSM returns to or remains in IDLE, deasserting valid_s and driving NOP on Link_Mgmt_Msg_Send and Message_receive.

The consuming_pending combinational wire captures the consume condition across all four FSM states: pending_rx_valid is true and the FSM is taking the pending-receive branch rather than the Message_send or Active_message_send branch. This single-wire formulation is required because the capture register has exactly one always_ff driver; assigning consume priority through a combinational intermediate prevents the multi-driver issue that would arise from two separate sequential blocks writing the same pending_rx_valid register.

Table 6-41: Message Handler Interface signals

Signal	Direction	Width	Description
lclk	Input	1	RDI domain clock
rst_n	Input	1	Active-low synchronous reset; returns FSM to IDLE and clears all output registers and the pending RX capture register
Message_send	Input	msg_no_e	Selected outgoing RDI message from unit_message_send_MUX; NOP = no pending transmission. Highest-priority input: non-NOP drives the FSM to LnkMsgS in any state
Active_message_send	Input	msg_no_e	Active-entry handshake message from wrapper_handshake_logic (RDI_ACTIVE_REQ or RDI_ACTIVE_RSP). Lowest-priority input: only accepted when Message_send == NOP and pending_rx_valid == 0
valid_r	Input	1	Received RDI link-management message valid strobe from Training_Mgmt (via SideBand_Top). Unconditionally latches Link_Mgmt_Msg_Received into the one-entry capture register on the rising clock edge
Link_Mgmt_Msg_Received	Input	msg_no_e	Received RDI link-management message number from Training_Mgmt; captured into pending_rx_msg when valid_r is asserted
valid_s	Output	1	RDI message TX valid to Training_Mgmt; asserted when the FSM is in LnkMsgS or ActvHsS with a non-NOP message to transmit
Link_Mgmt_Msg_Send	Output	msg_no_e	RDI link-management message number forwarded to Training_Mgmt for transmission over the sideband channel; NOP when valid_s is deasserted
Message_receive	Output	msg_no_e	Decoded received message presented to per-state modules (from the pending capture register); driven in LnkMsgR state, held at NOP in all other states

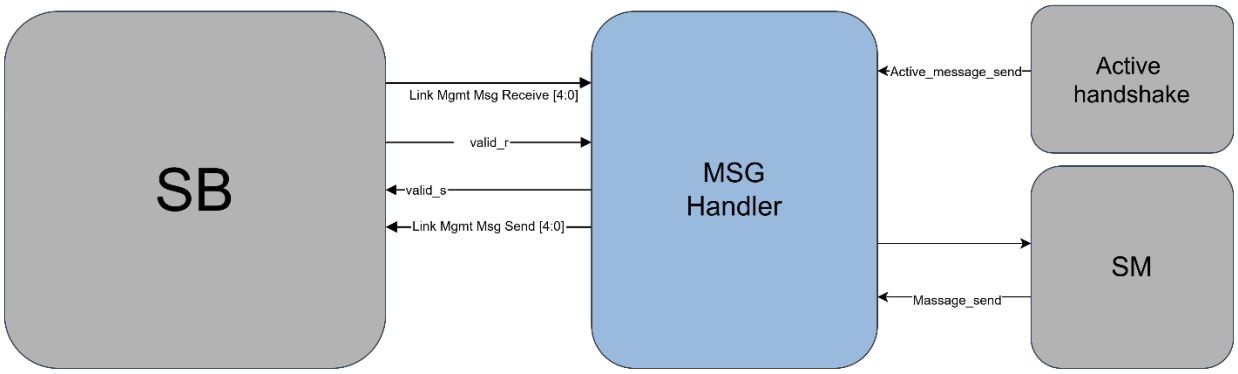


Figure 6-36: Message Handler diagram

7.0 Register File Design

7.1 Subsystem Overview

7.1.1 Role within the UCIe PHY System

Reg_File is the software-visible configuration and status subsystem of the UCIe PHY. It implements the complete UCIe Link DVSEC register block (Config Space) and the PHY-specific MMIO register block, enforcing per-field access attributes across all register types. On its output side it distributes software-programmed configuration values — link training start, target link width and speed, feature enable controls, clock mode and phase controls, the per-lane comparison mask, and the D2C error count thresholds — to MainSM for consumption during link initialization and training sequences. On its input side it receives two categories of hardware-driven inputs from MainSM: live status signals that are reflected directly into read-only register fields, and error-event capture inputs that are latched into the error log registers on the rising edge of pl_trainerror. Reg_File does not participate in mainband data transfer or sideband packet processing; it is a passive configuration and status store accessed exclusively through the sideband register access path.

7.1.2 Clock Domain and Sideband Access Path (clk_sb via Reg_Access)

Reg_File operates entirely in the clk_sb domain. It has no connection to the gated_lclk or lclk domains; all register reads, writes, and hardware status captures occur synchronously on the clk_sb clock edge. The clk_sb signal is generated inside SideBand_Top by the divide-by-eight stage that follows the sideband PLL, producing a 100 MHz clock that is forwarded through the UCIe_PHY wrapper to Reg_File. This makes Reg_File the only subsystem outside SideBand_Top that operates in the clk_sb domain.

The sole access path to Reg_File is the register-access bus driven by the Reg_Access block inside SideBand_Top. All register transactions — whether initiated locally by software through the on-die PCIe or CXL configuration mechanism or remotely through a sideband register-access packet from the remote die — arrive at Reg_File through this same bus. Reg_Access decodes the incoming 128-bit sideband packet, drives the address, byte enables, access width, and write data onto the bus, and asserts rd_en or wr_en. Reg_File responds with rf_rdata and rdata_vld for reads, or completes writes silently, and asserts addr_err_o when the presented address does not correspond to any implemented register.

The hardware status inputs from MainSM originate in the lclk domain. Because these values are quasi-static at the time they are consumed by Reg_File — they change only at well-defined training event boundaries, not cycle-by-cycle — no explicit synchronizer chain is inserted between the MainSM outputs and the Reg_File inputs. The pl_trainerror capture enable from the RDI_SM is similarly treated as a hardware write that occurs at a stable point in the training sequence, ensuring that the error log registers receive consistent values when captured.

7.2 Address Space Architecture

7.2.1 25-bit Address Bus Encoding

The register-access bus presents a 25-bit address to Reg_File on rf_addr. The encoding partitions the address into three fields with distinct decode semantics. Bit 24 is the space selector: when cleared to zero, the access targets the Config Space window (Config Space opcodes from Reg_DePacketizer produce this encoding); when set to one, the access targets the MMIO PHY block window (MEM and DMS opcodes produce this encoding). Bits 23:20 carry the four-bit Register Locator (RL) field, which identifies the target register block within the selected space. Bits 19:0 carry the byte offset within the selected block. This encoding is produced by Reg_DePacketizer directly from the sideband packet header fields, with the space selector derived from the packet opcode's type field (opcode[3]: 0 for CFG, 1 for MEM) and the remaining address fields taken directly from the packet header address field.

7.2.2 Config Space Window (addr[24]=0, 000h–023h)

When rf_addr[24] is zero, the access targets the Config Space window. This window holds the UCIe Link DVSEC register map beginning at byte offset 000h. The implemented single-module, single-locator configuration occupies a contiguous region of 36 bytes (nine 32-bit registers) from offset 000h through 023h. Within this range, the address decoder selects the target register by matching the low-order offset bits against the defined register offset table. Accesses to offsets within the Config Space window that do not correspond to any implemented register position are resolved as address errors, causing addr_err_o to be asserted.

7.2.3 MMIO PHY Block Window (addr[24]=1, 1000h–1108h)

When rf_addr[24] is one, the access targets the MMIO PHY register block. The implemented block spans offsets 1000h through 1108h, covering the PHY Capability, PHY Control, PHY Status, PHY Initialization and Debug, Training Setup 1 through 4, Current Lane Map Module 0, Error Log 0, Error Log 1, Runtime Link Test Control, and Runtime Link Test Status registers. Offsets outside this range within the MMIO window resolve as address errors. The MMIO block is discovered by software through the Register Locator pointer fields in the Config Space DVSEC structure.

7.2.4 Address Decode Logic (cfg_sel, phy_mmio_sel) and UR Error (addr_err_o)

Address decode within Reg_File is implemented as a two-level combinational decode. The first level uses rf_addr[24] to select between the cfg_mem byte array (Config Space) and the mmio_mem byte array (MMIO PHY block) as the target storage. The second level computes cfg_sel or phy_mmio_sel by comparing the low-order offset bits against the set of implemented register offsets. A match with a cfg_sel entry enables access to the corresponding 32-bit or 64-bit slice of cfg_mem; a match with a phy_mmio_sel entry enables access to the corresponding slice of mmio_mem. If neither selector matches — because the presented offset does not correspond to any implemented register position in the selected space — addr_err_o is asserted for one clock cycle, informing Reg_Access_FSM that the transaction should generate an Unsupported Request completion packet rather than a Successful Completion.

7.3 Config Space Register Map

7.3.1 PCIe Extended Capability Header and DVSEC Headers (000h–00Ah)

The Config Space window opens with the standard PCIe Extended Capability header structure required to make the UCIe Link DVSEC discoverable by software through the PCIe extended capability chain. Offset 000h holds the PCI Express Extended Capability Header containing the capability ID, capability version, and next-capability pointer. Offsets 004h and 008h hold DVSEC Header 1 and DVSEC Header 2, which carry the DVSEC vendor ID (0x1E98 for UCIe), the DVSEC ID that identifies this as the UCIe Link DVSEC capability, and the length field indicating the total byte extent of the capability structure. Offset 00Ah holds the second DVSEC header word. All fields in these registers have the RO attribute and are initialized from compile-time constants at integration; software may read them to identify the capability but cannot write to them.

7.3.2 UCIe Link Capability (00Ch) — RO + HWInit

The UCIe Link Capability register at offset 00Ch contains the advertised link capabilities of this PHY as they are made visible to software and to the remote die through the MBINIT.PARAM exchange. The register is composed of RO constant fields and HWInit fields. RO fields carry static constants that do not depend on integration context: the package type flag, the adapter format support bits for flit formats not supported in this implementation (all cleared), and reserved fields. HWInit fields are loaded from hardware input signals during domain reset and held read-only thereafter: hw_max_link_width_cap_i populates the Maximum Link Width field, hw_max_link_speed_cap_i populates the Maximum Data Rate field, and phy_advanced_pkg_cap_i populates the Advanced Package flag. Once reset is complete, software reads of HWInit fields return the strap values; writes are silently ignored.

7.3.3 UCIe Link Control (010h) — RW + HWInit

The UCIe Link Control register at offset 010h is the primary software-writable control register in the Config Space. RW fields in this register include the link training start bit, the target link width, the target link speed, the PMO enable, the PSPT enable (hardcoded zero in this implementation), and the L2SPD enable (hardcoded zero). HWInit fields in this register carry initial values loaded from hardware straps: for example, the x8 module width flag is initialized from the SPMW strap and held as a read-only indication once reset is complete, even though it occupies a bit position in the Control register. Software writes to RW fields update the stored values, which are subsequently driven on the corresponding control output ports (phy_start_ucie_link_training_ctrl_out, phy_target_link_width_ctrl_out, phy_target_link_speed_ctrl_out, phy_pmo_ctrl_out, phy_x8_width_mode_ctrl_out, and others) to MainSM.

7.3.4 UCIe Link Status (014h) — RO + RW1C + RW1CS (Mixed Attributes)

The UCIe Link Status register at offset 014h carries a mix of three attribute types. RO fields reflect live hardware status inputs driven by MainSM: the negotiated link width and link speed enable status fields (populated from reg_Link_Width_enable_status_i and reg_Link_Speed_enable_status_i), the PMO operational status (phy_sideband_pmo_status_i), the

clock mode and clock phase negotiation results, and the lane reversal indicator. These fields are continuously updated by the live hardware inputs and are never written by software. RW1C fields latch error event indicators that software must explicitly clear by writing a one to the relevant bit position: correctable and uncorrectable error detection flags are placed in this category. RW1CS fields behave identically to RW1C but retain their value across soft resets and retraining events, clearing only on a domain reset or an explicit software write-one-to-clear. Error detection bits that may persist across retraining cycles use the RW1CS attribute to preserve diagnostic information for software even after a recovery attempt.

7.3.5 Link Event and Error Notification Control (018h–01Ah)

Offsets 018h and 01Ah hold the Link Event Notification Control and Error Notification Control registers respectively. These registers contain RW fields that enable or disable the propagation of link event and error conditions as interrupts or sideband error messages. In this implementation, the interrupt vector number fields are read-only constants tied to zero, and the enable bits are RW but have no active interrupt generation logic connected to them, as interrupt generation infrastructure falls outside the defined scope of this digital PHY implementation. The registers are implemented with correct attribute enforcement so that software reads and writes behave as specified.

7.3.6 Register Locator (01Ch–020h)

The Register Locator register occupying offsets 01Ch through 020h provides the pointer by which software discovers the base address and type of the MMIO register blocks associated with this DVSEC capability. The Register BIR field identifies which PCIe Base Address Register is used to map the MMIO block into memory space. The Register Block Identifier field carries the encoding for the UCIe D2D/PHY Register Block (0h). The Register Block Offset Low and High fields together form the byte offset of the PHY register block from the base of the BAR. All fields in the Register Locator are RO and are initialized from compile-time constants at integration.

7.4 MMIO PHY Register Block

7.4.1 PHY Capability (1000h) — RO + HWInit

The PHY Capability register at MMIO offset 1000h advertises the physical-layer capabilities of this implementation. HWInit fields are loaded from the capability strap inputs during domain reset: phy_tx_vswing_encodings_cap_i populates the Supported TX Voltage Swing field, phy_rx_clk_mode_support_cap_i populates the RX Clock Mode Support field (initialized to 2'b10, declaring continuous clock mode support), phy_rx_clk_phase_support_cap_i populates the RX Clock Phase Support field (initialized to 2'b00, declaring differential-only operation), and phy_package_type_cap_i marks this as a Standard Package implementation. The phy_tarr_support_cap_i strap populates the TARR hardware support bit. RO constant fields in this register reflect capabilities not present in this implementation: TCM support and termination link capability are hardwired to zero.

7.4.2 PHY Control (1004h) — RW

The PHY Control register at MMIO offset 1004h is the central software-writable control register for the MMIO space. All fields are RW with active control output connections. The Initialization Control sub-field (bits 2:0) selects the LTSM training stop point, allowing software to pause training at MBINIT, MBTRAIN, RXDESKEW, or DATATRAINCENTER2 for debug purposes; encoding 000b selects the normal full-sequence bring-up to ACTIVE. The Resume Training bit (bit 5) triggers training resumption from the last paused state. Additional RW fields in this register control x8 width mode operation (phy_x8_width_mode_ctrl_out), the RX clock mode (phy_rx_clk_mode_ctrl_out), the RX clock phase (phy_rx_clk_phase_ctrl_out), TARR enable (phy_tarr_en_ctrl_out), and the PMO feature enable (phy_pmo_ctrl_out). All RW fields update their stored values on a byte-enable-qualified write and present those values combinationally on the corresponding control output ports.

7.4.3 PHY Status (1008h) — RO Live HW Inputs

The PHY Status register at MMIO offset 1008h contains only RO fields, all of which are continuously driven by live hardware inputs from MainSM rather than stored in the register array. Software reads of this register return the instantaneous state of the connected hardware signals at the time of the read access. The fields present the negotiated clock mode result (phy_clk_mode_status_i), the negotiated clock phase result (phy_clk_phase_status_i), the lane reversal indicator (phy_lane_rev_status_i), the TARR negotiation result (phy_tarr_status_i), the most-recently received I/Q correction parameter (phy_iq_correction_param_status_i), and the most-recently received EQ preset setting (phy_eq_preset_setting_status_i). Because these are live inputs rather than stored values, the read datapath for this register bypasses the mmio_mem byte array and selects the hardware input signals directly at the read address decode stage.

7.4.4 PHY Initialization and Debug (100Ch) — RW + RO

The PHY Initialization and Debug register at MMIO offset 100Ch contains a mix of RW and RO fields. The Initialization Control field (bits 2:0) and the Resume Training bit (bit 5) are RW and duplicate the training sequencing controls also present in the PHY Control register (their storage is shared or linked to the same internal state). A Tx Eq Status field (bit 4) is RO and reflects the current transmitter equalization state. Fields for which this implementation provides no analog equalization control are hardwired to zero. Writes to reserved RsvdP positions are silently ignored and reads return zero.

7.4.5 Training Setup Registers (1010h–1050h) — RW

Four training setup registers provide software-configurable parameters used by the LTSM during D2C point test and sweep sequences. Training Setup 1 (1010h) configures the TX data pattern selection, the valid pattern selection, the clock phase control, the training mode (continuous or burst), and the burst count. Training Setup 2 (1020h) configures the idle count and the iteration count for burst-mode pattern transmission. Training Setup 3 (1030h) is a 64-bit register carrying the 16-bit per-lane comparison mask, which is presented on the reg_lane_mask output and consumed by the LTSM to mask specific lanes from contributing to pattern comparator pass/fail

evaluation. Training Setup 4 (1050h) carries the 12-bit per-lane error count threshold and the 16-bit aggregate error count threshold, presented on cfg_max_err_thresh_perlane and cfg_max_err_thresh_aggr respectively. All fields in all four registers are RW with byte-enable-qualified write paths.

7.4.6 Current Lane Map Module 0 (1060h) — RW

The Current Lane Map Module 0 register at MMIO offset 1060h holds the current active lane map encoding for Module 0. The 3-bit lane map code follows the Standard Package encoding defined in the specification: 011b for x16 full width, 001b for x8 lower half, 010b for x8 upper half, 100b for x4 lower quarter, and 101b for x4 upper quarter. This register is RW, allowing software to read back the currently negotiated lane configuration and, through the Runtime Link Test Control register's Apply Module 0 Lane Repair bit, to trigger a width degradation at the next retrain cycle. The rt_apply_module_0_lane_repair_ctrl_out and module_0_lane_repair_id_ctrl_out outputs from Reg_File carry this software-programmed repair trigger to MainSM.

7.4.7 Error Log 0 (1080h) — ROS State Capture Shift Register (N, N-1, N-2)

The Error Log 0 register at MMIO offset 1080h captures LTSM state information at the moment of the first training error event following a domain reset. The register is organized as a first-event capture store with the ROS attribute: it latches its value on the first assertion of err_capture_en (pl_trainerror) and does not update on subsequent error events until a domain reset clears it. The captured fields are: the state-N code (bits 7:0) from err_state_capture (log0_state_n), the lane reversal flag (bit 8) from phy_lane_rev_err_log_i (log0_lane_reversal), the width degradation flag (bit 9) from phy_width_degrade_err_log_i (log0_width_degrade), the state N-1 code (bits 17:10) from log0_state_n_minus_1, and the state N-2 code (bits 25:18) from log0_state_n_minus_2. Together these three state codes provide a three-entry shift-register history of LTSM states leading up to the error, enabling software to reconstruct the training sequence context in which the failure occurred. Software writes to any field of an ROS register are silently discarded; only a domain reset returns the register to its default value of zero.

7.4.8 Error Log 1 (1090h) — ROS + RW1CS (Mixed) and State N-3

The Error Log 1 register at MMIO offset 1090h is a mixed-attribute register containing an ROS state field and four RW1CS event indicator bits. The ROS field (bits 7:0) captures the state N-3 code (log1_state_n_minus_3) at the first error event, extending the LTSM state history visible in Error Log 0 by one additional preceding state. The four RW1CS bits capture specific error context: bit 8 (State Timeout Occurred) is set by hardware when phy_state_timeout_i is asserted, indicating that a training state or sub-state timed out with a fatal outcome; bit 9 (Sideband Timeout Occurred) is set when phy_sb_timeout_i is asserted, indicating that an RDI sideband handshake did not complete within the 8 ms window; bit 10 (Remote LinkError Received) is set when phy_rm_link_err_i is asserted, indicating that the remote die requested a LinkError transition through the RDI sideband; and bit 11 (Internal Error) is set from phy_internal_err_i, which is tied to zero in this implementation. The RW1CS bits retain their set state across soft resets and

retraining cycles, clearing only on a domain reset or an explicit software write-one-to-clear, ensuring that error context remains visible after recovery attempts.

7.4.9 Runtime Link Test Control and Status (1100h–1108h)

Two registers support runtime link monitoring. The Runtime Link Test Control register at offset 1100h contains RW fields through which software may trigger a link width degrade at the next retrain cycle (Apply Module 0 Lane Repair bit, rt_apply_module_0_lane_repair_ctrl_out) and specify a lane repair ID (module_0_lane_repair_id_ctrl_out). Two RW/RO fields at bits 0 and 1 are implemented as RO with default value zero, consistent with the specification's guidance to implement these as RO for implementations that do not support the UCIe Test and Compliance register block. The Runtime Link Test Status register at offset 1108h contains an RO field that reflects the rt_link_busy_status_i input (the LTSM busy_flag), indicating to software when a link training or retraining operation is in progress.

7.5 Register Attribute Enforcement

7.5.1 RO — Hardwired Constants and Write-Ignore

Read-Only fields are implemented as wires directly connecting to either hardwired constant logic or to live hardware input ports; they have no write storage element. When a write transaction arrives at an address containing only RO fields, the wr_en strobe is presented to the register address decoder but no byte-array element is updated; the write completes silently with a Successful Completion returned to the requester, as required by the UCIe specification for writes to RO fields. When a read transaction targets an RO field, the read datapath selects the constant value or the live hardware input directly into the rf_rdata mux output.

7.5.2 HWInit — Capability Strap Sampling at Reset, Held RO Afterwards

HWInit fields are sampled from their hardware input signals during the assertion of rst_n and loaded into dedicated read-only storage elements that do not participate in the normal RW write path. Sampling occurs at the first rising clock edge after rst_n deasserts, capturing the strap values that are assumed stable before reset release. After domain reset completes, HWInit fields present as indistinguishable from RO fields to software: reads return the sampled strap value and writes are silently ignored. HWInit fields are not re-sampled during soft resets, retraining events, or link-state transitions; once captured at domain reset, the values are held until the next domain reset assertion.

7.5.3 RW — Byte-Enable Driven SW Write Path

Read-Write fields are stored in byte arrays (cfg_mem for Config Space, mmio_mem for MMIO space) and are updated by write transactions under byte-enable control. When wr_en is asserted, the register address decoder identifies the target register and computes which bytes of the 64-bit rf_wdata word correspond to the byte positions of the target register. For each byte position i where rf_be[i] is set, the corresponding byte of the addressed register is updated from rf_wdata[8*i+7:8*i]; bytes whose enable is deasserted retain their current value. This byte-

granular write path ensures that software can update individual fields within a register without disturbing adjacent fields in the same 32-bit or 64-bit access window. Reads of RW fields return the last value written by software, since no hardware logic modifies an RW field's stored value.

7.5.4 RW1C and RW1CS — HW Set Path, SW Write-1-to-Clear, Sticky Behavior

Write-One-to-Clear fields maintain a dedicated set-priority latch that hardware drives to one on the qualifying event. The HW set path takes priority over a simultaneous software write-one-to-clear operation, ensuring that no event is lost due to a timing coincidence with a software access. Software clears a set bit by writing a one to the corresponding field position in a write transaction; writing zero has no effect on a set bit. The RW1C variant clears on any reset, including soft resets initiated by the LTSM during retraining. The RW1CS variant clears only on a domain reset or an explicit software write-one-to-clear, preserving the event record across retraining cycles. In the UCIe Link Status register, error flags that should survive a retraining attempt use the RW1CS attribute, while event flags that are naturally cleared by a new training sequence use RW1C.

7.5.5 ROS — Error-Event Triggered Single Write, Reset-Only Clear

Read-Only Sticky fields implement a first-event capture mechanism. A dedicated capture-enable signal (err_capture_en, driven by pl_trainerror from the RDI_SM) gates an internal write enable to the ROS storage element. On the first assertion of err_capture_en following a domain reset, the associated hardware input value is latched into the ROS field and the write enable is suppressed for all subsequent assertions, preserving the first captured value. Software writes to ROS fields are unconditionally discarded. Only a domain reset clears the ROS field back to its default value of zero. This mechanism guarantees that the register file always holds the root-cause state information from the earliest training failure rather than overwriting it with data from later, possibly less informative, error events.

7.6 Read and Write Datapath

7.6.1 Byte-Granular Write Path (cfg_mem / mmio_mem byte arrays, rf_be enforcement)

The write datapath is built around two byte arrays: cfg_mem for the Config Space window and mmio_mem for the MMIO PHY block window. Each array is dimensioned to cover the full byte extent of its respective address window, with each element representing one byte of register storage. On a write transaction, the address decoder activates the relevant array slice, and the byte-enable vector rf_be selects which bytes within that slice are overwritten by the corresponding bytes of rf_wdata. Bytes in the target register that are associated with RO, HWInit, ROS, or RW1C/RW1CS attributes are excluded from the write path by masking their byte positions out of the byte-enable application, preventing software writes from altering hardware-controlled storage.

7.6.2 Registered Read Path (1-cycle latency, rdata_vld pulse)

All register reads incur exactly one clock cycle of latency. When rd_en is asserted, the address decoder combinationally selects the target register's data from the appropriate storage — the

cfg_mem or mmio_mem byte array for stored fields, direct hardware input signals for live RO fields, or the HWInit capture elements for capability fields. The selected 64-bit data word is registered into the read-data output register on the following clock edge, at which point rdata_vld is asserted for exactly one cycle and rf_rdata presents the valid data. The one-cycle read latency is fixed and applies to all access types; Reg_Access_FSM accounts for this latency by transitioning from EXECUTE to GEN only after rdata_vld is observed.

7.6.3 32-bit vs 64-bit Access (rf_is_64b_access)

Reg_File supports both 32-bit and 64-bit access granularities. The rf_is_64b_access flag, derived by Reg_DePacketizer from the high bit of the sideband packet opcode (opcode[3] = 1 for 64-bit opcodes), informs the read and write datapaths which granularity applies. For a 32-bit access, only the lower 32 bits of rf_rdata carry valid read data, and only the lower four bytes of rf_be are significant for writes. For a 64-bit access, the full 64-bit rf_rdata word is valid and all eight byte-enable positions may independently select bytes for update. The addr_err_o logic also accounts for access width when validating addresses, since some register offsets are defined as 32-bit only and a 64-bit access to those offsets may span into reserved address space.

7.7 HW Input and Control Output Interfaces

7.7.1 Capability Strap Inputs → HWInit Fields (Link Capability, PHY Capability)

Capability strap inputs are a set of compile-time constants driven into Reg_File from the UCIe_PHY top-level parameter boundary. They are presented as logic-level port inputs rather than Verilog parameters, allowing their values to be determined at integration time by the instantiating design rather than at elaboration time by the Reg_File module itself. At domain reset, these values are sampled and loaded into the HWInit storage elements of the UCIe Link Capability and PHY Capability registers. The straps cover: maximum link width (hw_max_link_width_cap_i, driven from MAX_LINK_WIDTH_CAP), maximum link speed (hw_max_link_speed_cap_i, from MAX_LINK_SPEED_CAP), package type (phy_advanced_pkg_cap_i, from ADVANCED_PKG_CAP — inverted for the package_type field), supported TX voltage swing (phy_tx_vswing_encodings_cap_i, from SUPPORTEDVSWING_CAP), clock mode support (phy_rx_clk_mode_support_cap_i, from CLK_MODE_CAP, default 2'b10), clock phase support (phy_rx_clk_phase_support_cap_i, from CLK_PHASE_CAP, default 2'b00), and TARR hardware support (phy_tarr_support_cap_i, from TARR_CAP).

7.7.2 Live HW Status Inputs → RO Fields (PHY Status, Link Status, Runtime Test Status)

Live hardware status inputs are driven from MainSM outputs and are presented directly to the read datapath of the registers they populate without passing through any storage element in Reg_File. The PHY Status register (1008h) is populated entirely from live inputs: phy_clk_mode_status_i reflects the negotiated clock mode, phy_clk_phase_status_i the negotiated clock phase, phy_lane_rev_status_i the current lane reversal state (log0_lane_reversal from LTSM), phy_tarr_status_i the TARR negotiation outcome, phy_iq_correction_param_status_i the most-recently received I/Q correction value, and phy_eq_preset_setting_status_i the most-recently

received EQ preset. The UCIe Link Status register (014h) similarly receives live inputs for the negotiated link width and speed enable status fields (reg_Link_Width_enable_status_i and reg_Link_Speed_enable_status_i from MainSM MBINIT.PARAM outcomes), the PMO operational status, the PSPT and L2SPD status bits, and the lane reversal indicator. The Runtime Link Test Status register (1108h) receives the LTSM busy_flag as rt_link_busy_status_i.

7.7.3 Error Event Inputs → ROS Capture (err_capture_en, err_state_capture, Error Logs)

The error log capture interface consists of a capture enable signal and a set of value inputs. The capture enable is err_capture_en, driven by pl_trainerror from the RDI_SM. On the first assertion of err_capture_en following a domain reset, the following values are latched into Error Log 0: err_state_capture (the current LTSM state-N code, log0_state_n), phy_lane_rev_err_log_i, phy_width_degrade_err_log_i, log0_state_n_minus_1, and log0_state_n_minus_2. Simultaneously, log1_state_n_minus_3 is latched into the ROS field of Error Log 1. The RW1CS event bits in Error Log 1 are set independently by their respective hardware input signals (phy_state_timeout_i, phy_sb_timeout_i, phy_rm_link_err_i, phy_internal_err_i) on each occurrence of the corresponding event, regardless of whether a prior capture has already occurred. The first-event capture mechanism in the ROS path and the sticky set mechanism in the RW1CS path together provide both a coherent first-error snapshot and an accumulation of all error types observed since the last domain reset.

7.7.4 SW Control Outputs (*_ctrl_out) → MainSM and Mainband

Control outputs from Reg_File carry software-programmed register field values to MainSM for consumption during link initialization, training, and operation. These outputs are combinational assignments directly from the RW storage elements of their source registers, presented as stable logic levels that MainSM samples at the appropriate points in the training sequence. The primary control outputs are: phy_start_ucie_link_training_ctrl_out (link training start bit, consumed by the RESET state to gate the exit to SBINIT); phy_target_link_width_ctrl_out and phy_target_link_speed_ctrl_out (target operating width and speed, consumed by MBINIT.PARAM); phy_x8_width_mode_ctrl_out (x8 mode control, forwarded to MBINIT.PARAM as the UCIe-S x8 parameter); phy_rx_clk_mode_ctrl_out and phy_rx_clk_phase_ctrl_out (clock mode and phase preferences, forwarded in the MBINIT.PARAM data payload); phy_pmo_ctrl_out (PMO software enable, determining whether the PMO bit is set in the SBFE request message); reg_lane_mask (per-lane comparison mask, used by the D2C point test in MBINIT.REPAIRMB and all MBTRAIN substates); and cfg_max_err_thresh_perlane and cfg_max_err_thresh_aggr (D2C error count thresholds, forwarded to the pattern comparator for all training sweeps).

8.0 Simulation, Verification and Hardware Validation

8.1 Verification Strategy Overview

8.1.1 Verification Levels (Unit, Wrapper, Integration, System)

The verification infrastructure is organized into four hierarchical levels, each building on the layer below. At the unit level, individual sub-blocks are exercised in isolation with directed stimulus applied directly to their ports; examples include the sb_serializer SVA module bound to the serializer under test, the unit_Timer_tb driving the dual-timer directly, and the unit_RX_D2C_PT_tb exercising the RX D2C point-test FSM with controlled pattern and compare signals. Isolating sub-blocks at this level allows rapid detection of functional errors before they propagate into higher-level assemblies.

At the wrapper level, a substate wrapper or protocol engine is tested as a unit, including all its internal sub-blocks and their interactions, while the surrounding context (mainband datapath, sideband channel) is modeled by lightweight behavioral stubs. The 13 MBTRAIN substate wrappers and the wrapper_D2C_PT testbenches fall into this category. At the integration level, multiple subsystems are connected and exercised jointly: the unit_mb_die2die_tb wires two complete mainband die models back-to-back, the wrapper_MBTRAIN_tb integrates the full MBTRAIN controller with all 13 substate wrappers and the shared D2C sweep engine, and the integration SideBand testbenches wire SideBand_Top with a complete link partner. At the system level, the full UCIe PHY — all four subsystems connected through UCIe_PHY — is exercised across the complete link initialization and active-transfer sequence.

The PHY was verified bottom-up in four layers, so that every failure could be localized to the smallest possible design unit before integrating upward:

Table 8-1: The four layers verification

Layer	What it proves	# of testbenches
Unit	Each leaf block in isolation (serializer, LFSR, mapper, comparator, FSM sub-states, RDI/SB packetizers, timers, handshakes)	91
Wrapper	Each leaf block in isolation (serializer, LFSR, mapper, comparator, FSM sub-states, RDI/SB packetizers, timers, handshakes)	29
Integration	Each leaf block in isolation (serializer, LFSR, mapper, comparator, FSM sub-states, RDI/SB packetizers, timers, handshakes)	16
System	The complete UCIe_PHY / UCIe_PHY_wrapper — two dies trained end-to-end over the real sideband and main-band datapaths	1(29 scenarios)

8.1.2 Simulation Environment (QuestaSim, Makefile/PowerShell Infrastructure)

All simulation is performed in Mentor QuestaSim (ModelSim). The simulation infrastructure supports both Linux and Windows development environments. On Linux, a Makefile at the repository root provides the primary entry point: testbenches are selected by the CONFIG variable (which identifies the listfile specifying the source file set) and the TOP variable (which names the top-level testbench module). The standard invocation is make run CONFIG=<config_name> TOP=<tb_module>; replacing run with debug opens the QuestaSim GUI with waveform display enabled. On Windows, a PowerShell script (run_sim.ps1) provides the equivalent interface: . un_sim.ps1 -CONFIG <config_name> -TOP <tb_module> -MODE run, with -MODE debug opening the GUI. A batch script (run_all_mbtrain_substates_tbs.ps1) automates sequential execution of all 13 MBTRAIN substate testbenches in a single invocation. Listfiles under sim/listfiles/ define the source file sets for each simulation configuration, separating the compilation list from the testbench control script and allowing the same RTL source to be shared across multiple testbench configurations.

8.1.3 Self-Checking Testbench Methodology (Pass/Fail Counters, Watchdog Timers)

All testbenches in this project are self-checking: they determine pass or fail autonomously without requiring manual waveform inspection. The primary mechanism is a pass/fail counter pair maintained throughout simulation. Correct behavioral outcomes increment the pass counter; detected errors increment the fail counter. At the end of simulation, the testbench reports the total counts and asserts a PASS or FAIL banner. A global watchdog timer provides a secondary safety mechanism: if the device under test fails to produce an expected output or complete a required handshake within a predefined number of clock cycles, the watchdog fires, terminates simulation, and reports a timeout failure. This prevents simulations from hanging indefinitely on a deadlocked protocol state.

For MBTRAIN verification, dedicated testbench infrastructure files (ltsm_tb_if.sv and ltsm_tb_attachments.sv) define a shared testbench interface and a set of reusable checker attachments that can be bound to any substate wrapper under test. These attachments monitor the substate's sideband message output, mainband control output, and done/error handshakes, comparing observed behavior against expected sequences derived from the scenario specification. The same infrastructure is reused across all 13 substate wrapper testbenches and the top-level wrapper_MBTRAIN_tb.

8.2 Mainband Verification

The mainband verification campaign covers the complete digital_mb datapath at multiple levels, from individual lane processing blocks through full die-to-die loopback, culminating in a training-sequence integration test that exercises the LTSM-driven control of the datapath across all MBINIT sub-states.

8.2.1 SER/DES Loopback Wrapper with Frame-Lock Scoreboard (unit_tx_deser_wrapper_tb)

The unit_tx_deser_wrapper_tb connects the serializer output of the TX datapath directly to the deserializer input of the RX datapath, forming a closed-loop serial loopback without a physical channel. A frame-lock scoreboard monitors the recovered parallel data against the expected data produced by locally running a synchronized copy of the LFSR, flagging any deserialized word that deviates from the expected PRBS-23 sequence. This testbench validates the serialization and deserialization timing, confirms that the bit-counter alignment in the deserializer produces correct word boundaries, and verifies the CDC synchronization path between the received-clock domain and the parallel clock domain by checking that rdata_vld pulses are produced at the correct cadence.

8.2.2 Full TX→RX Loopback with Demapper Verification (unit_tx_demap_wrapper)

The unit_tx_demap_wrapper testbench extends the loopback to include the full RX path including LFSR_RX descrambling and the Demapper. Protocol data words of known content are injected at the lp_data input of the mapper, serialized, deserialized, descrambled, and reassembled by the Demapper. The testbench compares the reconstructed o_out_data against the original lp_data input, confirming that the full data integrity is preserved across the complete transmit-receive pipeline for both x16 and x8 width configurations. The valid framing path is verified in parallel by confirming that o_pl_valid is asserted at the correct boundaries relative to the data words.

8.2.3 Framed Path and Stream Path Integration (unit_mb_framed_path_tb, unit_mb_path_stream_tb)

Two integration-level testbenches exercise the mainband datapath in the context of its framing and streaming behavior. The unit_mb_framed_path_tb applies a sequence of back-to-back protocol data transfers and verifies that the 4-high, 4-low valid framing envelope is correctly generated and detected, that the 8-UI block alignment rule is respected across consecutive transfers, and that the receiver correctly identifies frame boundaries and presents byte-aligned data to the upper layers. The unit_mb_path_stream_tb exercises continuous streaming data transfer across the complete mapper-to-demapper path, measuring throughput and confirming that no data words are dropped or corrupted across an extended burst sequence.

8.2.4 Die-to-Die Integration Testbench with Fault Injection (unit_mb_die2die_tb)

The unit_mb_die2die_tb instantiates two complete unit_mb_die models and crosses their serial interfaces — the transmit outputs of die 0 connect to the receive inputs of die 1 and vice versa — forming a functional die-to-die loopback. The testbench runs a 5×5×3 parametric sweep over data patterns, operating widths, and LFSR states to verify that all combinations of transmit and receive configuration are correctly handled. Fault injection is applied by deliberately introducing bit errors at the serial interface and confirming that the pattern comparator in the receiving die detects and counts the errors correctly, that per-lane pass/fail vectors reflect the injected fault locations, and that aggregate error counts match the expected totals.

8.2.5 Full Mainband Training Sequence Verification (unit_mb_train_seq_tb)

The unit_mb_train_seq_tb drives the complete MBINIT-phase training sequence, exercising the mainband datapath under LTSM-style sequential control through the REPAIRCLK, REPAIRVAL, REVERSALMB, and REPAIRMB sub-states in order. The testbench applies the training control signals in the sequence defined by the specification, verifies that the done pulses from the pattern comparator and valid comparator are produced at the correct points, confirms that the clock and track detection flags are asserted when the corresponding pattern is present, and verifies that the lane reversal detection logic correctly identifies and reports a reversed physical lane order when the per-lane ID pattern is received in non-sequential order.

8.3 Sideband Verification

The sideband verification campaign uses SystemVerilog Assertions bound to the production RTL for the SerDes components, and directed behavioral testbenches for the higher-level protocol behavior including PMO mode.

8.3.1 sb_serializer SVA Formal Properties (packet length, gap enforcement, clock forwarding)

The sb_serializer_sva module is bound to the sb_serializer production RTL using the SystemVerilog bind construct, making the assertions active during all simulation configurations that include the serializer. Four properties are checked. The p_valid_hold property verifies data stability: when tx_data_valid is asserted but tx_rdy is not, the parallel input data must remain stable on the next clock edge, confirming that the caller correctly holds valid data until the buffer accepts it. The p_S_SHIFT_length property verifies packet duration: when the serializer transitions into the S_SHIFT state (without PMO enabled), the state must last exactly 64 serial clock cycles, which is disabled during PMO mode since consecutive packets may be loaded without an intervening idle. The p_S_GAP_length property verifies gap enforcement: when the serializer enters S_GAP, the state must last exactly 32 serial clock cycles regardless of PMO mode. The p_clock_forward property verifies forwarded clock gating: whenever the serializer is not in S_SHIFT, TXCKSB must be zero, confirming that the clock is suppressed during idle and gap intervals as required by the UCIe specification.

8.3.2 sb_deserializer SVA Formal Properties (counter wrap, packet-done position, CDC pulse)

The sb_deserializer_sva module is bound to the sb_deserializer production RTL. Six properties are checked. The p_counter_wrap property verifies bit-counter rollover: when bit_cnt equals DATA_WIDTH-1 (63), the counter must be zero on the next RXCKSB cycle. The p_counter_increment property verifies monotonic counting: when bit_cnt is not at its maximum value, it must increment by exactly one on the next RXCKSB cycle. The p_packet_done_position property verifies boundary alignment: the packet_done signal may only be asserted when bit_cnt equals DATA_WIDTH-1, confirming that completion is signaled only at the 64th bit. The p_packet_done_pulse property verifies pulse behavior: packet_done must be deasserted on the

cycle following its assertion, preventing spurious double-completion. The p_data_vld_pulse property verifies the CDC output: the rx_data_vld output in the parallel clock domain must be deasserted on the cycle following its assertion. The p_data_stability property verifies output stability: when rx_data_vld is asserted, the rx_parallel_data_out value must remain stable on the subsequent cycle, preventing the data word from changing while it is being consumed.

8.3.3 PMO Mode Behavioral Verification

The PMO behavioral testbench verifies the back-to-back transmission behavior introduced by Performant Mode Operation. With pmo_en asserted, the testbench applies a sequence of consecutive valid 64-bit words to the serializer and confirms that successive S_SHIFT phases are entered without any intervening S_GAP phase, producing a continuous 64-UI-per-packet output stream. The testbench also verifies backward compatibility on the receive side: the deserializer is driven with a mixture of gap-present and gap-absent packet streams (as would occur during the PMO activation window defined by the specification) and the testbench confirms that correct 64-bit words are produced by rx_data_vld pulses regardless of whether an inter-packet gap is present or absent between the two 64-bit serial sequences.

8.4 LTSM and D2C Verification

LTSM verification covers the D2C point-test engine used by MBINIT.REPAIRMB and all MBTRAIN substates, all 13 MBTRAIN substate wrappers individually, the MBTRAIN top-level integration, and an OOP class-based testbench suite for systematic scenario coverage of the full MBTRAIN sequence.

8.4.1 Unit-Level D2C Point Test Testbenches (unit_RX_D2C_PT_tb, unit_TX_D2C_PT_tb)

Two unit-level testbenches verify the local and partner D2C point-test FSMs in isolation. The unit_TX_D2C_PT_tb drives the TX-initiated D2C point test FSM, applying the sideband message handshake sequence (Start req/resp, LFSR clear req/resp, results req/resp, End req/resp) and verifying that the FSM drives the correct mainband TX pattern control signals at each stage, that it correctly asserts local_test_d2c_done at the appropriate handshake boundary, and that per-lane pass and aggregate pass outputs are correctly derived from the injected comparison result inputs. The unit_RX_D2C_PT_tb exercises the RX-initiated counterpart, verifying the symmetric receive role: the FSM enables the pattern comparator at the correct time, collects comparison results, and reports the outcome through the partner_test_d2c_done path after the result request-response exchange completes.

8.4.2 D2C Point Test Wrapper Verification (wrapper_D2C_PT_tb, wrapper_D2C_PT_top_tb — 100 Randomized Iterations)

The wrapper_D2C_PT_tb verifies the integration of the local and partner D2C FSMs within the shared wrapper_D2C_PT module, which manages the MUX between MBINIT and MBTRAIN configuration inputs and routes mainband control signals between the two FSMs depending on which test is active. The wrapper_D2C_PT_top_tb extends this verification to 100 randomized test

iterations, varying the pattern configuration, error threshold, lane mask, and injection of simulated per-lane pass/fail results across each iteration. Randomization covers all four test roles (local TX-initiated, partner TX-initiated, local RX-initiated, partner RX-initiated) and confirms that the wrapper correctly arbitrates the sideband TX output between the two FSMs under all active role combinations.

8.4.3 MBTRAIN Sub-state Directed Testbenches (all 13 sub-states)

Each of the 13 MBTRAIN substate wrappers has a dedicated directed testbench. All 12 substates other than LINKSPEED are marked as Completed and Verified in the repository; the LINKSPEED testbench is marked as Completed. Each testbench drives the substate enable signal, applies the expected sideband message stimulus from the simulated remote die, monitors the substate's sideband TX output and mainband control output against the expected sequence, and checks that the done signal is asserted after the correct handshake completion. Error paths are also covered: each testbench injects a simulated error condition (a timeout, a failed comparison result, or an unexpected message) and confirms that the substate asserts its error output rather than incorrectly asserting done. The full set of covered substates is VALVREF, DATAVREF, SPEEDIDLE, TXSELFCAL, RXCLKCAL, VALTRAINCENTER, VALTRAINVREF, DATATRAINCENTER1, DATATRAINVREF, RXDESKEW, DATATRAINCENTER2, LINKSPEED, and REPAIR.

8.4.4 MBTRAIN Top-Level Integration Testbench (wrapper_MBTRAIN_tb)

The wrapper_MBTRAIN_tb integrates unit_MBTRAIN_ctrl with all 13 substate wrappers and the shared D2C sweep engine and verifies the complete nominal golden path through the MBTRAIN sequence: VALVREF → DATAVREF → SPEEDIDLE → TXSELFCAL → RXCLKCAL → VALTRAINCENTER → VALTRAINVREF → DATATRAINCENTER1 → DATATRAINVREF → RXDESKEW → DATATRAINCENTER2 → LINKSPEED → LINKINIT. It also covers the L1-exit entry path (re-entry at SPEEDIDLE) and a subset of the speed-degrade and width-degrade routing arcs from LINKSPEED. The testbench applies a behavioral sideband agent that responds to each substate's sideband request messages with the appropriate response, advancing the sequencer through the handshake protocol at each substate boundary. The implementation is complete and functional; additional scenario coverage for the full matrix of degrade, retrain, and error paths is deferred to the class-based suite described in Section 8.4.5.

8.4.5 Class-Based OOP MBTRAIN Testbench Suite

The class-based MBTRAIN testbench suite (wrapper_MBTRAIN_class_based_tb) applies an object-oriented testbench architecture to provide systematic scenario coverage, a behavioral reference model, and functional coverage tracking for the full MBTRAIN sequence. The suite is organized into nine cooperating SystemVerilog class files.

8.4.5.1 Testbench Architecture (Config, Driver, Monitor, Scoreboard, Coverage, SB Agent)

The top level (mbtrain_cb_tb_top.sv) instantiates the design under test (wrapper_MBTRAIN) and the testbench environment (mbtrain_cb_env), connecting them through the shared testbench interface (mbtrain_cb_if.sv). The environment contains six cooperating components. The

configuration object (mbtrain_cb_config.sv) parameterizes each test scenario: it specifies the entry point (VALVREF, SPEEDIDLE, TXSELFCAL, or REPAIR), the per-substate behavioral outcome (success, speed degrade, width degrade, PHY retrain request, or error), the DTC1 loopback count for RXDESKEW, and the active lane configuration used by the sweep engine. The driver (mbtrain_cb_driver.sv) consumes transaction objects and translates them into timed signal stimulus on the testbench interface, applying sideband response messages at the correct protocol points and advancing the mainband comparison result inputs. The monitor (mbtrain_cb_monitor.sv) observes the DUT outputs and assembles observed behavior into transaction objects that are forwarded to the scoreboard and coverage collector. The scoreboard (mbtrain_cb_scoreboard.sv) compares observed transactions against expected values produced by the configuration object and increments the pass or fail counter accordingly. The sideband agent (mbtrain_cb_sb_agent.sv) handles the sideband message handshake protocol independently of the main driver, responding to each substate's SB TX request with the correct SB RX response at the correct time. The test library (mbtrain_cb_testlib.sv) defines the set of named test scenarios that the top-level testbench selects and executes.

8.4.5.2 Transaction Model and D2C Behavioral Reference Model (mbtrain_cb_d2c_model)

The transaction model (mbtrain_cb_transaction.sv and mbtrain_cb_types_pkg.sv) defines the data types used to communicate between the testbench components. A transaction object encapsulates a single substate execution event: the substate identifier, the configured behavioral outcome, the sideband messages exchanged, the mainband control signals observed, and the done or error assertion. The D2C behavioral reference model (mbtrain_cb_d2c_model.sv) implements a software model of the D2C point-test and sweep engine behavior, producing the expected per-lane pass vectors and sweep result codes for any given configuration of error thresholds, lane mask, and injected bit-error count. The driver uses the reference model to determine which comparison result inputs to drive at each point in the training sequence, and the scoreboard uses it to predict the expected done condition and lane map outcome after the REPAIR substate.

8.4.5.3 Functional Coverage (mbtrain_cb_coverage)

The functional coverage collector (mbtrain_cb_coverage.sv) defines covergroups that track the cross-product of covered scenario dimensions. Coverage points include the MBTRAIN entry point (four values), the LINKSPEED exit decision (LINKINIT, SPEEDIDLE, REPAIR, PHYRETRAIN), the DTC1 loopback count (0 through 4 iterations), the width degradation encoding applied by REPAIR (all five lane map codes), and the combined speed-then-width degrade path. The coverage database is reported at the end of simulation, identifying scenario combinations that have not yet been exercised by the current test suite. The outstanding coverage gaps inform the list of known incomplete scenarios described in Section 8.7.2.

8.5 RDI_SM Verification

The RDI_SM verification campaign covers the dual-timer, individual per-state modules, and the handshake and clock-gating behavioral paths.

8.5.1 unit_Timer_tb — 1µs and 16ms Timer Verification

The unit_Timer_tb verifies both timer outputs of unit_Timer independently. For the 1 µs timer, the testbench holds start_time_1us high for a fixed number of cycles equal to the calculated T1US_LIMIT at the configured clock frequency (default 2 GHz, producing a limit of 1999 cycles) and confirms that time_1us is asserted exactly on the cycle when the counter reaches zero. The testbench then deasserts start_time_1us and verifies that the counter resets and time_1us deasserts on the same cycle. For the 16 ms timer, an equivalent test is applied with start_time_16ms, verifying that time_16ms asserts after the full 16 ms count and clears immediately when the enable is removed. Both timers are also tested for independence: simultaneous assertion of both enables is confirmed to produce both outputs at their respective boundaries without interference.

8.5.2 RDI State Sub-module Directed Testbenches

Each RDI per-state module (unit_reset_state, unit_active_state, unit_active_pmnak_state, unit_retrain_state, unit_L1_state, unit_L2_state, unit_linkreset_state, unit_linkerror_state, unit_disabled_state) has a dedicated directed testbench. Each testbench applies the state's enable signal and drives the required input conditions — lp_state_req, message_receive, lp_linkerror, pl_error, and the relevant timer and handshake done signals — to confirm that the state's next_state output transitions correctly under each defined exit condition. For the Active state, the stall path is also covered: stall_req is confirmed to be asserted when pl_error or a PM state request is received, and stall_done is confirmed to deassert stall_req correctly. The Active_PMNAK state's behavior when a stall is requested while a power management NAK is active is also verified to produce the correct next_state transition.

8.5.3 Stall Handshake and Clock Gating Behavioral Verification

The stall handshake testbench verifies the four-state FSM in unit_stall_handshake by driving stall_req through the IDLE → STALLREQ → STALLACK → STALLDONE sequence with controlled assertion and deassertion of lp_stallack. The pl_stallreq output is confirmed to be high in both STALLREQ and STALLACK, and stall_done is confirmed to be asserted only in STALLACK. The mapper-stall latch (stall_done_latched) is verified to be set when stall_done is asserted and cleared when mapper_en falls to zero, confirming the level-hold behavior required for MainSM's mapper-enable gating path. The clock gating testbench verifies unit_gating_logic by driving pl_state_sts through the gating-permitted states (RESET, LinkReset, Disabled) and confirming that lclk_g is deasserted in each, and through Active to confirm that lclk_g is asserted. The training-start window behavior is also verified: with phy_start asserted and sticky_sb_pattern_detected unset, lclk_g is confirmed to remain asserted through the RESET state to keep the mainband clock running during the SBINIT training-start window.

8.6 Register File Verification

The register file verification campaign confirms correct attribute enforcement, address decode behavior, and access-width handling through directed testbenches applied to the Reg_File module.

8.6.1 Attribute Enforcement Testing (RO write-ignore, RW1C clear, ROS capture)

The attribute enforcement testbench covers the five register attribute types implemented in Reg_File. For RO fields, the testbench applies write transactions to known RO bit positions and confirms that subsequent reads return the original values unchanged, verifying the write-ignore behavior. For HWInit fields, the testbench applies reset and confirms that the stored value equals the hardware input strap value, then applies a write transaction and confirms the value is not modified. For RW fields, the testbench writes known data patterns under full and partial byte enables and confirms that read-back returns the written values, with unwritten bytes retaining their previous content. For RW1C and RW1CS fields, the testbench asserts the hardware set path while simultaneously applying a software write-one-to-clear, confirming that the set takes priority and the field remains set rather than being cleared. A subsequent write-one-to-clear is then applied and the field is confirmed to clear. For ROS fields, the err_capture_en is asserted twice with different err_state_capture values, and the testbench confirms that the stored value is the one from the first assertion and is not overwritten by the second, verifying the first-event capture behavior.

8.6.2 Address Decode and UR Error Response Testing

The address decode testbench covers the full address space of both the Config Space window (addr[24]=0) and the MMIO PHY block window (addr[24]=1). For each implemented register, the testbench applies a read transaction at the register's byte offset and confirms that rf_rdata is valid and rdata_vld is asserted one cycle later with no addr_err_o. For unimplemented offsets within both windows and for addresses that lie entirely outside the defined ranges, the testbench confirms that addr_err_o is asserted for one cycle, rdata_vld is not asserted, and rf_rdata is held at zero. The boundary conditions are specifically tested: the last implemented register offset, the first unimplemented offset immediately following it, and a fully out-of-range address at the extremes of the 25-bit address space.

8.6.3 64-bit vs 32-bit Access and Byte-Enable Verification

The access width testbench applies both 32-bit and 64-bit read and write transactions to registers that support each access type and confirms correct behavior in both cases. For 32-bit accesses (rf_is_64b_access deasserted), the testbench confirms that only the lower 32 bits of rf_rdata carry valid register data and that write transactions update only the lower four byte positions within the register. For 64-bit accesses (rf_is_64b_access asserted), the full 64-bit word is confirmed to be read or written correctly. The byte-enable verification applies individual byte-enable patterns (one hot, alternating, and all-ones) to 32-bit and 64-bit write transactions and confirms that exactly the enabled bytes are updated while the remaining bytes retain their previous values. Byte-enable patterns that span an RO field are also tested to confirm that the RO byte is not modified even when its byte-enable bit is set in the transaction.

8.7 System-Level Verification of the UCIe PHY

8.7.1 Purpose and scope

The system-level testbench is the top of the verification pyramid. Where unit and wrapper tests prove blocks in isolation, the system bench proves that the complete PHY behaves as a real chiplet link: two independent dies, powered up from reset, configured only through their register interface, negotiating a link with each other over the real serial wires, and exchanging data. Nothing is short-circuited — the two dies only ever see each other through the physical pins.

The bench lives in UCIe_PHY_wrapper_tb.sv (≈2150 lines) and drives the UCIe_PHY_wrapper DUT, which is the pin-identical digital+analog reassembly of the PHY.

8.7.2 Two-die "silicon-realistic" topology

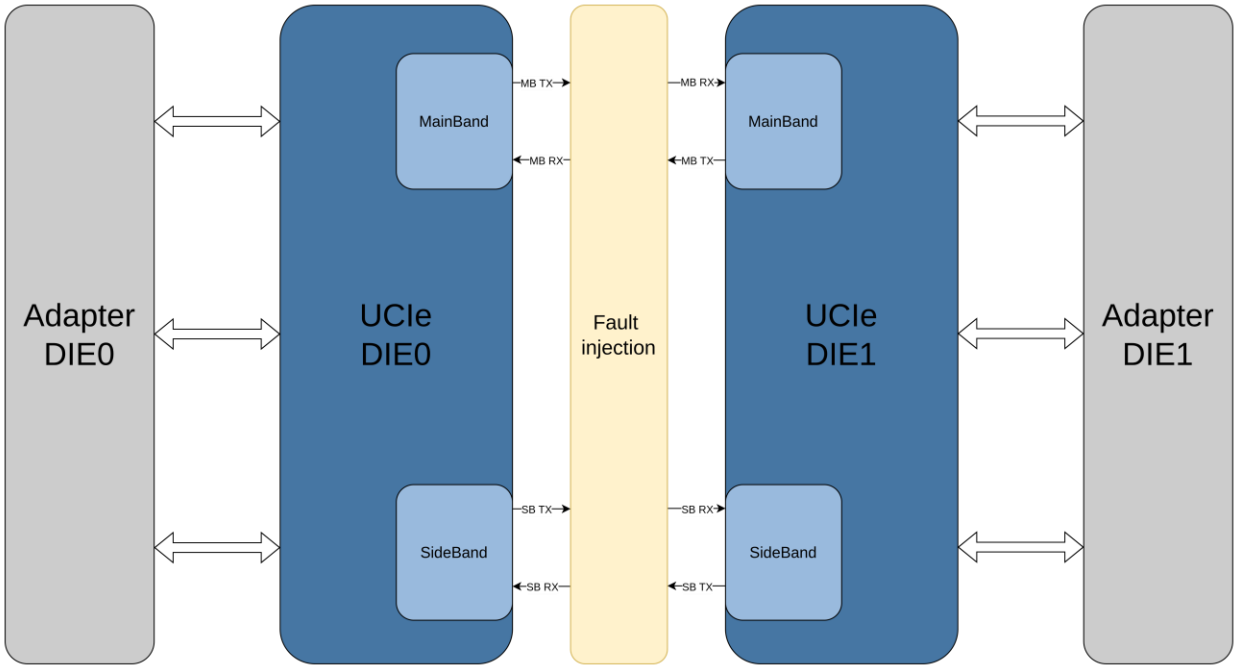


Figure 8-1: UCIe system level TB Architecture

Two full DUT instances are created, distinguished only by a MODULE_ID parameter (die-0 = master/initiator, die-1 = follower):

The dies are wired crosswise: die-0's transmit lanes feed die-1's receive lanes and vice-versa, for all four main-band signal groups (16 data lanes, valid, differential forwarded clock P/N, and the track lane) plus the two-wire sideband (clock + data). This is what makes the test a true link bring-up rather than a loopback.

8.7.3 The channel model — where faults are injected

The cross-connect is not a plain wire; it is a behavioral channel that the scenarios use to inject the physical impairments UCIe training is designed to survive (lines 100–122):

Table 8-2: Types of Injected Error

Knob	Effect	Exercises
corrupt_0to1 / corrupt_1to0	Forces selected lanes stuck-at-0 in either direction	Lane repair, width degradation
reverse_lanes_0to1 / reverse_lanes_1to0	Bit-reverses lane order on the wire	Lane-reversal detection & correction
rx_vld_error_inject_0_to_1	XOR-flips the valid lane	Valid-frame error handling / PHYRETRAIN
block_sideband	Zeroes both sideband directions	Watchdog / TRAINERROR path

Because these live in the channel, the DUT has no knowledge of them — it must detect and recover using only its spec-defined training mechanisms, exactly as it would against real silicon defects.

8.7.4 Configuration through the real register path

Nothing is force-programmed. Each die is brought up by streaming sideband register transactions into its Reg_File, exactly as the adapter would:

build_wr_header() assembles a proper UCIe sideband request header (opcode, dstid=LOCAL_PHY, srcid=ADAPTER, address, byte-enables, control parity).

send_chunks64/32() serialize the header+payload as 32-bit lp_cfg chunks on the sideband access clock.

Helpers reg_wr_cfg, reg_wr_mem, reg_rd_cfg, reg_rd_mem cover both CFG and MMIO spaces and read-back with completion capture (receive_pl_cfg_packet).

Bring-up programs three registers per die — Training Setup 4 (comparator thresholds), PHY Control (Rx-clock mode/phase, force-x8), and UCIe Link Control (target width, target speed, and the Start-UCIe-Link-Training bit, asserted on die-0 only). The follower die joins through the sideband handshake. This means the config/decode path is verified end-to-end alongside the datapath.

8.7.5 Driving the RDI (adapter) face

The bench plays the role of the adapter above each PHY:

Bring-up sequencing (do_bringup_custom): program both dies, then when each reaches LOG_LINKINIT, request Active on the RDI state interface; wait until both dies report LOG_ACTIVE and pl_state_sts == Active.

Handshake responders: lp_clk_ack/lp_stallack are made to follow pl_clk_req/pl_stallreq in clocked logic, modeling a compliant adapter's clock- and stall-handshake (lines 184–191).

PM / error stimuli: lp_state_req, lp_wake_req, lp_linkerror drive L1/L2 entry, wake, and link-error scenarios.

8.7.6 Checking — scoreboard, data proof, and observability

Scoreboard. A single self-checking primitive chk(cond, msg, sc) increments a global fails counter and tags every result with its scenario number; the run ends with a top-level RESULT: PASS / FAIL (N failing checks) verdict, so the whole 29-scenario regression is one pass/fail signal.

Full-duplex data proof. do_data_transfer() is the functional payload check: after ACTIVE, die-0 drives a DEADBEEF flit and die-1 drives CAFEBABE; the test passes only when each die receives the other's pattern on its pl_data output. Because the flit travels mapper → scrambler → serializer → channel → deserializer → descrambler → demapper at the negotiated width and speed, a correct compare proves the entire main-band datapath end to end — not just that the FSM reached ACTIVE.

White-box observability. Hierarchical probes expose the three states that matter for debugging training: the LTSM macro-state (current_ltsm_state_n), the internal RDI state that drives it (rdi_state_sts), and the lagging RDI status output (pl_state_sts). Any change auto-prints a timestamped snapshot per die. print_active_status() and print_ctrl_regs() dump the negotiated width, speed, TX/RX lane masks, active deser lanes, and the lane-reversal / width-degrade flags read back from the DUT — giving each scenario a self-documenting evidence trail for the thesis.

8.7.7 Making long timeouts simulatable

Two timescale concessions keep runtime tractable while staying spec-proportional:

The RDI 1 µs / 16 ms timers are shrunk via defparam (CLK_FREQ = RDI_CLK_FRQ).

The 8 ms LTSM watchdog is left real and is speed-aware in the RTL itself (derives its cycle count from the negotiated PLL speed), so it fires at a true 8 ms at any speed with no TB hack. Because that makes it dominate runtime, the watchdog scenario (SC2) is gated off by default (RUN_SC2 = 0) and enabled only when specifically exercising TRAINERROR.

8.7.8 Regression control

All scenarios are individually gated by the enabled_scenarios[1..29] bit-mask (lines 569–597), so the thesis can present either the full 29-scenario regression (all bits set, the default) or an isolated scenario for a focused waveform figure. The bench is invoked through the project's standard flow — make run CONFIG=UCIe_PHY TOP=UCIe_PHY_wrapper_tb — which builds and runs it under QuestaSim.

8.7.9 Coverage-of-intent — what the 29 scenarios span

Table 8-3: The 29 system-level scenarios

#	Scenario	What it verifies
1	Happy Path → ACTIVE	Full bring-up: SB register programming → training → both dies ACTIVE → data transfer
2	Watchdog Timeout (SB blocked)	Sideband blocked → 8 ms watchdog fires → TRAINERROR entry
3	Asymmetric Width Negotiation (x16 vs x8 → x8)	Dies request different widths; link negotiates the common minimum
4	Symmetric Lane Reversal + Retry	Both dies reverse package routes; reversal detection & correction
5	Asymmetric Lane Reversal	Only one die reverses; one-sided reversal handling
6	Width Degradation — fault lanes 8–15 (Die 0)	Faulty upper lanes → degrade x16 → x8 with retry
7	Width Degradation — fault lanes 0–7 (Die 1)	Faulty lower lanes on the far die → degrade + retry
8	Width Degradation — both dies faulted (0–7 / 8–15)	Independent faults on each die → common degraded width
9	Link-speed mismatch (20 GT/s vs 16 GT/s)	Both dies request different speeds → negotiate common speed
10	PM mode L1 entry + wake	ACTIVE → L1 low-power entry → wake → re-train to ACTIVE
11	L2 entry → RESET → re-train	Deepest PM state, exit through RESET, full recovery
12	TRAINERROR (rdi = LinkError) → clear → recover	Error injected from RDI, cleared, link recovers
13	Valid-lane boundary error injection	Error on the valid framing lane; detection & handling
14	MBTRAIN lane degradation (REPAIR mid-train)	Lane fault during MBTRAIN triggers the REPAIR flow
15	MBTRAIN speed degradation (speed-dependent all-lane fault)	All lanes fail at speed → speed step-down during training
16	Target settings > hardware capabilities	Control-register request exceeds HW caps → clamps to caps
17	Lane reversal + asymmetric width degrade (x16 → x8)	Reversal combined with width degrade
18	Asym width negotiate (x16 vs x8 → x8) + degrade to x4, **with** reversal	Two-stage width reduction with reversal
19	Asym width negotiate (x16 vs x8 → x8) + degrade to x4, **without** reversal	Two-stage width reduction, no reversal
20	Speed degradation (1 step) on degraded x4-upper link	Speed step-down on an already width-degraded link
21	Speed degradation (2 steps) on degraded x4-upper link	Two speed steps on the degraded link
22	Lane reversal + 1-step speed degradation	Combined reversal and speed degrade
23	Asym width degrade (x16 → x8) + 1-step speed degrade	Width + one speed step
24	Asym width degrade (x16 → x8) + 2-step speed degrade	Width + two speed steps

25	Lane reversal + width degrade (x16 → x8) + 1-step speed degrade	Triple combination (case A)
26	Lane reversal + width degrade (x16 → x8) + 1-step speed degrade	Triple combination (case B)
27	Speed degradation until TRAINERROR	Repeated speed degrade exhausts options → TRAINERROR
28	Valid-frame error during ACTIVE → PHYRETRAIN → TXSELFCAL	Runtime error in ACTIVE forces a re-train loop back through TXSELFCAL
29	Sideband completion & clock handshake	Register-write completion packet + pl_clk_req/lp_clk_ack handshake during a read

The 29 scenarios are not ad-hoc; together they form a coverage grid over the four training axes that UCIe defines, plus power and error management:

- Bring-up & negotiation (SC1, 3, 9, 16) — happy path, asymmetric width, speed mismatch, capability clamping.
- Lane reversal (SC4, 5) — symmetric and asymmetric.
- Width degradation / repair (SC6–8, 14) — faults on upper/lower/both dies, mid-train REPAIR.
- Speed degradation (SC15, 20, 21, 27) — 1-step, 2-step, and degrade-to-exhaustion → TRAINERROR.
- Combined-axis stress (SC17–19, 22–26) — reversal × width × speed in every meaningful combination, the hardest cases for the negotiation logic.
- Power management (SC10 L1, SC11 L2→RESET).
- Runtime error & recovery (SC12 RDI LinkError, SC13 valid-lane error, SC28 ACTIVE→PHYRETRAIN→TXSELFCAL, SC2 watchdog).
- Sideband integrity (SC29 register-completion + clock handshake).

8.8 Verification Results Summary

8.8.1 Test Completion Status per Subsystem

The following table summarizes the completion and verification status of the testbench suite across all five subsystems.

Table 8-4: Test Completion Status

Subsystem	Testbench	Scope	Status
Mainband	unit_tx_top_tb, MB_TX_TOP_tb	TX datapath unit	Completed & Verified
Mainband	unit_tx_deser_wrapper_tb	SerDes loopback + scoreboard	Completed & Verified

Subsystem	Testbench	Scope	Status
Mainband	unit_tx_demap_wrapper_tb	TX→RX loopback + Demapper	Completed & Verified
Mainband	unit_mb_framed_path_tb, unit_mb_path_stream_tb	Framed and stream path	Completed & Verified
Mainband	unit_mb_die2die_tb	Die-to-die 5×5×3 sweep + fault injection	Completed & Verified
Mainband	unit_mb_train_seq_tb	Full training sequence	Completed & Verified
Sideband	sb_serializer_sva (4 properties)	SerDes SVA formal	Completed & Verified
Sideband	sb_deserializer_sva (6 properties)	SerDes SVA formal	Completed & Verified
Sideband	PMO behavioral	Back-to-back + backward compat	Completed & Verified
LTSM / D2C	unit_TX_D2C_PT_tb, unit_RX_D2C_PT_tb	D2C unit FSMs	Completed & Verified
LTSM / D2C	wrapper_D2C_PT_tb, wrapper_D2C_PT_top_tb	D2C wrapper (100 rand. iters)	Completed & Verified
LTSM / D2C	wrapper_VALVREF_tb … wrapper_REPAIR_tb (12 of 13)	MBTRAIN substate directed	Completed & Verified
LTSM / D2C	wrapper_LINKSPEED_tb	MBTRAIN LINKSPEED directed	Completed & Verified
LTSM / D2C	wrapper_MBTRAIN_tb	MBTRAIN top-level integration	Completed & Verified
LTSM / D2C	wrapper_MBTRAIN_class_based_tb	OOP class-based suite	Completed & Verified
RDI SM	unit_Timer_tb	Dual timer (1 µs + 16 ms)	Completed & Verified
RDI SM	Per-state directed TBs (9 states)	RDI state sub-modules	Completed & Verified
RDI SM	Stall + clock-gating TBs	Handshake and gating behavior	Completed & Verified
Register File	Attribute enforcement TB	RO/HWInit/RW/RW1C/ROS	Completed & Verified
Register File	Address decode and UR TB	Config + MMIO windows	Completed & Verified
Register File	Access width + byte-enable TB	32-bit / 64-bit / BE	Completed & Verified

8.9 FPGA Prototype and Hardware Validation

8.9.1 Motivation and approach

The PHY was taken from RTL to a running FPGA prototype on a Zynq UltraScale+ (Zynq MPSoC) platform to demonstrate the digital design in real hardware and to build a software-driven bring-up flow. The obstacle is fundamental: the PHY's analog hard macros — PLL, SerDes, and tri-state drivers — cannot map to FPGA fabric. The strategy is therefore:

Strip the analog macros, keep the entire digital PHY, and fold every transmit boundary back onto its own receive boundary as a self-loopback. A processor then configures the PHY over the sideband, trains it to ACTIVE, streams data through the main band, and compares transmitted vs. received.

This proves the whole digital stack — MainBand scramble/descramble + framing, Sideband packetization + register access, the RDI state machine, and LTSM training — in silicon-representative conditions, without needing the analog front end.

8.9.2 FPGA-target RTL

Two things had to be re-created because the removed hard macro used to provide them:

Loopback fabric (digital_ucie_loopback.sv) folds the parallel MainBand and Sideband boundaries back on themselves with no SerDes latency — TX and RX share one word clock, so the RX descrambler stays word-locked to the TX scrambler. On-chip word-clock gate — under `ifdef FPGA the core's clock-enable drives a BUFGCE (global, glitch-free) to regenerate the gated MainBand clock the analog PLL used to supply.

8.9.3 Clocking and constraints

The design runs fully synchronous (the wrapper ties clk_sb = lclk and feeds lclk as both the gated word clock and the PLL clock). The constraints file documents the modelled clock relationship (matching the simulation), and clock-domain crossings are handled by the AXI-Stream FIFOs and handshakes, with the RX clock-detector sample registers marked ASYNC_REG + false-path.

Table 8-5: Clock Domains

Clock	Freq	Role
pll_clk	2000 MHz (model)	analog bit-rate model — **must be re-scaled for real silicon**
lclk / gated_lclk	125 MHz	MainBand word clock
clk_sb	100 MHz	Sideband parallel clock

8.9.4 Embedded system

The PHY IP is integrated into a Vivado IP-Integrator block design with the processing system of an Avnet Ultra96-V2 board — a Xilinx Zynq UltraScale+ MPSoC (ZU3EG, SBVA484).

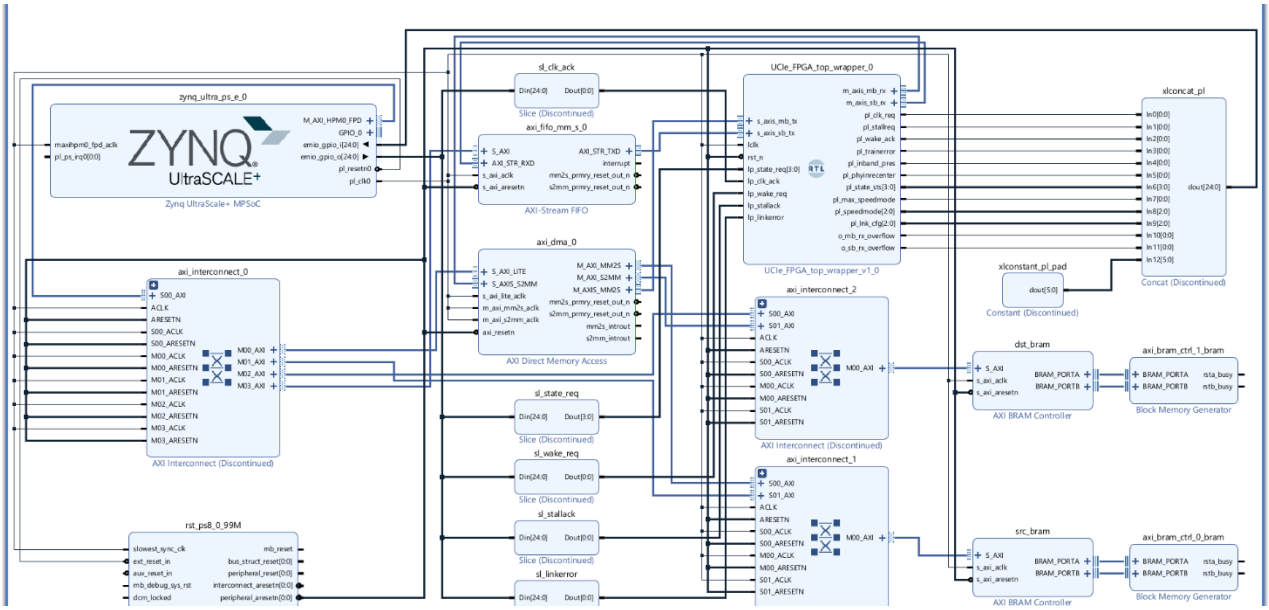


Figure 8-2: Vivado IP-Intergrator block design

The PS talks to the PHY over three hardware channels, chosen to match each traffic type:

Table 8-6: Embedded System Hardware Channels (UCIe PHY)

Channel	Xilinx IP	Purpose
MainBand data	AXI DMA (XAxiDma) via BRAM	bulk 512-bit flit transfer TX↔RX
Sideband config	AXI Stream FIFO (XLlFifo)	register read/write packets + completions
RDI control/status	PS EMIO GPIO (XGpioPs, bank 3)	drive lp_*, observe pl_*

A fixed address map (single source of truth in ucie_driver.h, taken from the Vivado Address Editor) places the DMA, FIFO, and the TX/RX BRAMs in the PS memory space, and an RDI EMIO bit-map assigns the low GPIO bits to the RDI request/status lines. The block design builds to a clean bitstream, exported as a hardware handoff (system_wrapper.xsa) to the Vitis platform.

8.9.5 Bare-metal application and driver

A Vitis bare-metal C application (vitis_ucie/app_component/src/, ~64 KB text) drives the full bring-up. The driver (ucie_driver.c/.h) abstracts the three channels; main.c runs the sequence:

Init — bring up DMA, sideband FIFO, and EMIO GPIO. Start training — program the PHY registers (Link Control, PHY Control, Training Setup) and assert Start UCIe Link Training over the sideband.

Bring up to ACTIVE — drive the RDI state request via GPIO and poll pl_state_sts until ACTIVE (2 s budget). Read back the negotiated width/speed from the Link Status register. MainBand DMA loopback — fill the source BRAM, DMA into the PHY, loop through, DMA out to the destination BRAM. Compare TX vs. RX and print TEST RESULT: PASS/FAIL. Sideband register access reuses the same UCIe opcodes and register offsets as the RTL, so the on-target software and the simulation exercise the identical configuration path. A lightweight event monitor decodes RDI-status / FIFO-occupancy changes to a one-liner so a stalled bring-up is immediately localizable.

8.9.6 Results

- Lint (Verilator): clean of correctness defects; residual items are cosmetic style only.
- Digital loopback simulation: the stripped digital PHY trains and passes in self-loopback.
- Vivado / Zynq: the block design builds to a clean bitstream, exported to the Vitis platform.
- On-target: the bitstream and bare-metal application were deployed to the physical Ultra96-V2 board and ran the full sequence — Init → training → ACTIVE → MainBand DMA loopback → TX/RX compare — printing TEST RESULT: PASS over UART, with the negotiated link width/speed read back from the status register. This demonstrates the complete digital PHY operating on real silicon.

9.0 Conclusion

9.1 Summary of Work

9.1.1 Design Objectives Revisited

This project set out to deliver a structurally complete, specification-grounded digital implementation of the UCIe Standard Package physical layer, scoped to four objectives: a mainband datapath supporting the full byte-to-lane mapping, scrambling, training pattern, lane reversal, and width degradation functions of the specification; a sideband subsystem operating on a permanently powered auxiliary domain with Sideband Feature Extensions and Sideband Performant Mode Operation; a main state machine integrating the Link Training State Machine and the RDI state machine with full coverage of all normative states and sub-states; and a register file enforcing every UCIe-defined access attribute across the Link DVSEC and PHY-specific register spaces. The implementation presented across Chapters 4 through 7 satisfies all four objectives at the RTL level, with each subsystem instantiated as a peer block beneath the UCIe_PHY integration wrapper and connected through the explicit inter-subsystem interfaces documented in Chapter 3.

The analog abstraction boundary established at the outset of the project — in which all PLL, SerDes, phase-interpolator, and voltage-reference-DAC functionality is represented to the digital subsystems exclusively through parallel-domain control and result signals — was maintained consistently across all four subsystems. unit_mb_die, SideBand_Top, MainSM, and Reg_File communicate with their respective analog hard macros only through this defined boundary, with no subsystem modeling or depending on the internal behavior of the analog circuitry. This consistency confirms that the stated scope constraint of a digital-only implementation was achieved without compromise at any subsystem boundary.

9.1.2 Summary of Each Subsystem Implementation

The mainband datapath (unit_mb_die, Chapter 4) implements the complete transmit and receive parallel-domain pipeline: the Mapper and Demapper perform byte-to-lane distribution and reassembly across x16, x8, and x4 width configurations; LFSR_TX and LFSR_RX implement the specification's 23-bit scrambling polynomial in both PRBS-23 pattern and per-lane ID modes; the pattern comparator and valid comparator implement per-lane and aggregate error accumulation against configurable thresholds; and the clock pattern generator and detector implement both 128-UI burst mode and continuous embedded-clock mode for the forwarded clock. The datapath is structured as a pure consumer of control signals from MainSM, with no autonomous sequencing logic of its own, confirming the architectural separation between datapath and control established in Chapter 3.

The sideband subsystem (SideBand_Top, Chapter 5) implements the complete packet-level protocol engine on the clk_sb domain: the Link_Controller performs priority arbitration, 128-to-64-bit packetization through sb_mapper and sb_demapper, and SBINIT pattern generation and detection through sb_pattern_engine and sb_pattern_detector; Training_Mgmt bridges the lclk-

domain LTSM and RDI message sources across asynchronous FIFOs into the clk_sb domain; RDI_control manages the bidirectional adapter configuration channel with credit-based flow control; and Reg_Access implements the register-access endpoint with a four-state FSM and UCIe-compliant completion generation. The sb_serializer and sb_deserializer correctly implement the mandatory 32-UI inter-packet gap in baseline mode and the gap-free back-to-back transmission required by Sideband Performant Mode Operation, verified by the SVA properties described in Chapter 8.

The main state machine (MainSM, Chapter 6) integrates LTSM_TOP and RDI_SM as coordinated peers. The LTSM implementation covers all ten LTSM controller states — RESET, SBINIT, MBINIT, MBTRAIN, LINKINIT, ACTIVE, PHYRETRAIN, L1, L2, and TRAINERROR — with the MBINIT phase further decomposed into its six sub-states (PARAM, CAL, REPAIRCLK, REPAIRVAL, REVERSALMB, REPAIRMB) and the MBTRAIN phase into its thirteen sub-states, including the shared D2C sweep engine reused across MBINIT.REPAIRMB and all MBTRAIN substates that require a pattern sweep. The §4.5.3.8 two-way TRAINERROR-entry handshake is correctly differentiated between states requiring sideband coordination (MBINIT, MBTRAIN, LINKINIT) and states where a direct transition is appropriate (SBINIT, PHYRETRAIN). The RDI state machine implements the full nine-state RDI protocol with the AWAKE, STALL, ACTIVE ENTRY, and CLK handshakes, coarse clock gating coordinated with the training-start window, and the Link-Error two-way handshake with correct initiator/responder collision resolution.

The register file (Reg_File, Chapter 7) implements the complete Config Space DVSEC register block and the PHY-specific MMIO register block with correct enforcement of all five UCIe access attributes — RO, HWInit, RW, RW1C/RW1CS, and ROS — verified at the bit level by the attribute-enforcement testbenches described in Chapter 8. The 25-bit address decode correctly partitions the Config Space and MMIO windows and generates Unsupported Request completions for unmapped addresses, and the byte-granular write path correctly supports both 32-bit and 64-bit access granularities under independent byte-enable control.

9.2 Contributions

9.2.1 Spec-Compliant LTSM Covering All Normative States and Sub-states

The LTSM implementation covers all eight top-level normative states defined by UCIe Specification Revision 3.0 — RESET, SBINIT, MBINIT, MBTRAIN, LINKINIT, ACTIVE, PHYRETRAIN, and TRAINERROR — together with the L1 and L2 power-management states, the six MBINIT sub-states, and the thirteen MBTRAIN sub-states. Each state and sub-state is implemented as an independently testable RTL module with an explicit done/error interface to its parent sequencer, and the shared D2C point-test sweep engine is reused consistently across every sub-state that requires a pattern sweep, avoiding duplicated training logic. The non-linear transition arcs defined by the specification — the L1-to-SPEEDIDLE re-entry, the PHYRETRAIN-negotiated re-entry point selection among TXSELFCAL, SPEEDIDLE, and REPAIR, and the four-way LINKSPEED exit decision — are all implemented in conformance with the specification's defined sequencing rules.

9.2.2 Mainband Datapath with x8 Width Mode, Width Degradation, and Lane Reversal

Beyond the baseline x16 full-width operating mode, the mainband datapath implements the optional x8 Width Mode feature, with byte-to-lane mapping correctly supporting both the lower-half and upper-half x8 configurations as negotiated during MBINIT.PARAM. Width degradation logic in MBINIT.REPAIRMB and MBTRAIN.REPAIR correctly negotiates and applies reduced lane configurations when a fault is detected in a contiguous portion of the active lane set, and lane reversal detection in MBINIT.REVERSALMB correctly identifies and corrects a reversed physical lane order using per-lane ID pattern comparison. These three features together demonstrate that the datapath is not limited to the simplest operating configuration but correctly implements the specification's fault-tolerance and flexibility mechanisms.

9.2.3 Sideband Subsystem with PMO and SFES

The sideband subsystem implements two specification-optional features in addition to the baseline protocol: Sideband Feature Extensions, which negotiates the extended feature set through the two-stage SBFE request/response exchange during MBINIT.PARAM, and Sideband Performant Mode Operation, which doubles sideband throughput by eliminating the inter-packet gap once negotiated. The PMO implementation correctly handles the activation-timing asymmetry defined by the specification, in which the receiver must tolerate both gap-present and gap-absent packet streams during the transition window, as verified by the behavioral testbench described in Section 8.3.3.

9.2.4 RDI State Machine with Full Clock Gating and Stall Handshake

The RDI state machine implements coarse clock gating of the mainband parallel-domain clock during RESET, LinkReset, Disabled, and the power-management states, correctly coordinating the gating window with the SBINIT training-start sequence so that the mainband clock remains available during the period when sideband pattern detection is establishing the inter-die channel. The stall handshake correctly implements the latched mapper-stall mechanism, ensuring that the mainband mapper is held off for the full duration of an adapter stall acknowledgment cycle without requiring the LTSM to track stall state directly. This decoupling between the RDI state machine's run-time power and flow-control contract and the LTSM's training sequencing logic reflects the architectural separation defined by the specification between these two coordinating state machines.

9.2.5 Register File with Full Attribute Enforcement

The register file implements all five register access attributes defined by the UCIe specification with bit-accurate behavior: RO fields are correctly write-ignored, HWInit fields are correctly sampled once at domain reset and held thereafter, RW fields support byte-granular partial writes, RW1C and RW1CS fields correctly prioritize the hardware set path over a simultaneous software clear and correctly differentiate sticky persistence across soft resets, and ROS fields correctly implement first-event capture semantics that preserve the earliest training failure context rather than being overwritten by subsequent error events. This attribute-correct implementation spans both the Config Space DVSEC register block and the PHY-specific MMIO register block in their entirety.

9.2.6 Multi-Level Verification Infrastructure

The verification campaign spans four levels — unit, wrapper, integration, and system — applying self-checking testbenches with pass/fail counters and watchdog timeouts at every level, SystemVerilog Assertions bound directly to the production sideband SerDes RTL, and an object-oriented class-based testbench suite for the MBTRAIN sequence providing configuration-driven scenario generation, a behavioral D2C reference model, and functional coverage tracking. This multi-level structure allowed each of the four digital subsystems to be verified both in isolation and in the context of die-to-die and full-system integration, with a dual-OS simulation infrastructure (Makefile and PowerShell) supporting development on both Linux and Windows.

9.2.7 FPGA Prototype Demonstrating Synthesis Feasibility at 125 MHz

The complete digital PHY — all four subsystems integrated beneath the UCIe_PHY wrapper — was synthesized and prototyped on an FPGA platform, achieving a maximum operating frequency of 125 MHz with no timing violations. A C-based embedded test infrastructure exercised register read/write correctness, HWInit strap sampling, ROS first-event capture, and RW1CS sticky persistence directly in hardware, providing physical confirmation of behavior that was otherwise validated only in simulation. This result demonstrates that the RTL implementation is synthesizable as a coherent whole and not merely simulatable as isolated unit-level blocks.

9.3 Limitations

9.3.1 Excluded Features (PSPT, L2SPD, TARR, Compliance Testing, Multi-module)

Five features defined by UCIe Specification Revision 3.0 were excluded from the implementation scope by explicit design decision: Priority Sideband Packet Transfer, L2 Sideband Power Down, Tx Adjustment during Runtime Recalibration, PHY Compliance Testing Support, and Multi-module Configuration. The corresponding capability and status bits for these features (PSPT, L2SPD, TARR enable, and the compliance test register block) are present in the register file's port list as required by the register map structure but are either hardwired to their default disabled value or left without connected control logic, as documented in Chapter 7. These exclusions were scoped at project outset to bound the implementation to a single-module Standard Package configuration with a tractable verification surface; vendor-defined extensions to any of these mechanisms were likewise excluded throughout.

9.3.2 Analog Components Not Modeled (PLL, AFE, PI, Vref DAC)

The implementation is strictly digital. The PLL, the analog front-end transmit and receive circuitry, the phase interpolators used for clock and data centering, and the voltage-reference DACs used for receiver calibration are represented to the digital subsystems only through their parallel-domain control and result interface, as defined by the analog abstraction boundary in Chapter 3. No circuit-level, transistor-level, or electrical behavior of these components has been modeled, synthesized, or verified within this project. The training algorithms that exercise these interfaces — the D2C sweep engine in particular — are verified against behavioral models that produce

plausible pass/fail and sweep-result patterns, not against any physically accurate representation of an actual phase interpolator's sweep characteristics or a voltage-reference DAC's settling behavior.

9.3.3 Single-Module Scope Only

All four subsystems and the UCIe_PHY integration wrapper are structured around a single Standard Package module. The lane-map encoding used throughout MBINIT.REPAIRMB, MBTRAIN.REPAIR, and the Current Lane Map register in Reg_File assumes exactly one module's worth of lanes (16 data lanes plus the valid, clock, and track lanes of a single module), and no module-selection or inter-module arbitration logic exists anywhere in the design. This scope decision is consistent with the explicit exclusion of Multi-module Configuration from the project's feature list and means the implementation, as it stands, cannot be directly extended to a multi-module Standard Package or any Advanced Package configuration without additional architectural work.

9.4 Future Work

9.4.1 Advanced Package Variant Extension (x64/x32, Lane Repair, Redundant Sideband)

Extending the implementation to the UCIe Advanced Package would require support for the wider x64 and x32 lane-count configurations, true per-lane repair (as opposed to the contiguous-half/quarter width degradation implemented for the Standard Package), and the redundant sideband lane pairing that the Advanced Package permits for sideband fault tolerance. The mainband Mapper, Demapper, and lane-map encoding logic would need to be generalized beyond the fixed 16-lane assumption that pervades the current REPAIRMB and REPAIR sub-state implementations, and the sideband subsystem would need additional logic to manage the redundant lane pair and failover decision.

9.4.2 Multi-module Configuration Support

Supporting multiple Standard Package modules within a single link would require a module-selection and arbitration layer above the current single-module UCIe_PHY wrapper, replication of the per-module training state where the specification requires independent per-module negotiation, and aggregation logic for the lane-map and error-log results that are currently scoped to one module. The MBINIT and MBTRAIN sub-state controllers would need to be extended or replicated to manage the per-module training sequence while preserving a coherent top-level LTSM state for the link as a whole.

9.4.3 PSPT and L2SPD Feature Integration

Integrating Priority Sideband Packet Transfer would require extending the Link_Controller's TX priority arbiter and sb_mapper to support a third priority tier above the existing training/adapter distinction, together with the negotiation logic in MBINIT.PARAM's SFES exchange. Integrating L2 Sideband Power Down would require extending the L2 state's exit and entry handling in both the LTSM and RDI_SM to coordinate sideband channel power-down and wake-up with the existing mainband L2 handling, along with the corresponding negotiation bit in the SBFE exchange. Both

features currently have their capability and status register fields present but tied to their disabled default, as noted in Section 9.3.1, providing a ready integration point for this future work.

9.4.4 Full-System Co-simulation with a D2D Adapter BFM

The current verification campaign exercises the RDI adapter-facing interface signals at the signal level through directed and behavioral testbenches, but no complete bus-functional model of a D2D Adapter has been integrated to drive the full UCIe_PHY wrapper through a realistic protocol-layer traffic pattern. Developing such a BFM would allow end-to-end verification from the FDI-equivalent adapter boundary through the physical serial pins, exercising the stall, wake, and active-entry handshakes under adapter-driven rather than testbench-driven timing and providing a more representative confidence level for system integration.

9.4.5 ASIC Synthesis and Timing Closure

The FPGA synthesis result reported in Section 8.8.4 demonstrates synthesizability at 125 MHz on an FPGA target but does not constitute ASIC timing closure. Future work would involve technology-specific synthesis and place-and-route with a real or representative analog IP front-end, timing closure at the target operating frequencies defined by the UCIe specification's supported data rates, and power analysis of the clock-gating and PMO mechanisms implemented in this design under realistic switching activity.

9.4.6 UCIe-3D Package Variant Exploration

The UCIe-3D package variant, which targets direct die-to-die bonding with a substantially different electrical and physical interface than the Standard or Advanced Package bump-based interconnect, was outside the scope considered in this project. Exploring a UCIe-3D-targeted variant of this digital implementation would require revisiting the analog abstraction boundary itself, since the parallel-domain interface assumptions made for a bump-based SerDes front-end may not directly translate to the very different signaling and density characteristics of a 3D-bonded interconnect.

تمهيد

أدى الانتشار السريع لبنى "النظام داخل الحزمة (SiP) "المفككة والقائمة على تقنية "الرقاقات الفرعية (chiplets) " إلى خلق طلب على واجهات ربط قياسية ومفتوحة بين الرقاقات(die-to-die) ، تكون قادرة على توفير نطاق ترددي عالٍ، وزمن انتقال منخفض، وقابلية للتشغيل البيني بين تقنيات سيليكون متنوعة. ويلبي معيار Universal Chiplet" Interconnect Express" (UCIe) هذه الحاجة من خلال تحديد حزمة اتصالات طبقية متكاملة للأنظمة القائمة على الرقاقات الفرعية. وتستعرض هذه الأطروحة بنية العتاد الرقمي وتنفيذ الطبقة الفيزيائية (PHY) الخاصة بمعيار UCIe، بما يتوافق مع المراجعة ٣,٠ من مواصفات المعيار.

يقتصر نطاق هذا العمل على المجال الرقمي؛ إذ يتم التعامل مع المكونات التناظرية — بما في ذلك حلقات قفل الطور (PLLs)، ودوائر الواجهة الأمامية التناظرية(AFE) ، ومستنبطات الطور(PIs) ، ومحولات الإشارة الرقمية إلى تناظرية (DACs)الخاصة بمرجع الجهد — (Vref) من خلال واجهات التحكم والحالة الرقمية الخاصة بها، مما ينتج عنه تصميم قابل للتوليف (synthesizable) بالكامل ومستقل عن أي تقنية محددة للملكية الفكرية (IP) التناظرية.

تتألف البنية المنفذة من خمسة أنظمة فرعية رئيسية. يقوم مسار بيانات النطاق الرئيسي (Mainband) بتنفيذ وظائف نقل البيانات، والتشويش(scrambling) ، وتعيين المسارات(lane mapping) ، وعكس المسارات، وتقليل العرض (width degradation)، ودعم التدريب، وذلك لكل من عرضي التشغيل x16 و .x8ويوفر النظام الفرعي للنطاق الجانبي (Sideband) واجهة اتصال تسلسلي متكاملة، تشمل توجيه الحزم، وعمليات الوصول إلى السجلات، وامتدادات ميزات النطاق الجانبي(SFES) ، وتشغيل النطاق الجانبي في وضع الأداء العالي .(PMO) كما تغطي آلة حالة تدريب الرابط (LTSM) الشاملة جميع إجراءات التهيئة والمعايرة والتدريب وإعادة التدريب واستعادة النظام من الأخطاء المحددة في معيارUCIe ، وذلك عبر جميع حالات المستوى الأعلى الثماني والحالات الفرعية الثلاث عشرة لتدريب النطاق الرئيسي. وتوفر آلة حالة الواجهة الخام بين الرقاقات (RDI) الحد الفاصل القياسي بين الطبقة الفيزيائية (PHY)ومحول الربط بين الرقاقات، بينما توفر مجموعة السجلات (Register File) المتوافقة مع المعايير وظائف التهيئة، ومراقبة الحالة، والتحكم في التدريب، وتسجيل الأخطاء، وإدارة الرابط أثناء التشغيل.

تم التحقق من صحة البنية المنفذة باستخدام منهجية محاكاة هرمية تغطي الأنظمة الفرعية الخمسة، مما أثبت صحة التشغيل السلوكي عبر سيناريوهات تهيئة الرابط، والتدريب، وحالات التشغيل النشط.

يستثني التصميم كلاً من: نقل حزم النطاق الجانبي ذات الأولوية(PSPT) ، وإيقاف تشغيل طاقة النطاق الجانبي في المستوى(L2SPD) L2 ، وتعديل الإرسال (Tx) أثناء إعادة المعايرة في وقت التشغيل(TARR) ، ودعم اختبارات الامتثال للطبقة الفيزيائية(PHY) ، وتهيئة الوحدات المتعددة، والامتدادات المحددة من قبل الموردين. تستند جميع القرارات المعمارية بشكل صارم إلى المتطلبات الإلزامية (المعيارية) الواردة في المراجعة ٣,٠ من مواصفات .UCIe وحسب علم المؤلفين، يُعد هذا العمل واحداً من أوائل التطبيقات الرقمية الموثقة علناً والقائمة على المواصفات لطبقة الربط المادي (PHY) الخاصة بحزمة معيار UCIe (وفقاً للمراجعة ٣,٠)؛ مما أثمر عن بنية معمارية معيارية وقابلة للتحقق، ومناسبة للاستخدام كوحدة ملكية فكرية (IP block) لطبقة PHY قابلة لإعادة الاستخدام ضمن تصميمات "النظام داخل الحزمة (SiP) "غير المتجانسة والقائمة على تقنية الرقاقات الجزئية.(chiplets)

UCIe الحزمة القياسية

الطبقة الفيزيائية

التصميم والتحقق من الوظيفة

بناءاً علي مواصفات ال UCIe مراجعة ٣,٠ إصدار ١,٠

إعداد:

عبدالله محمد صلاح الدين

عبدالرحمن عدوي علي

محمد عادل عبدالعاطي

محمدأنور عبداللطيف

محمد السيد إبراهيم

مؤمن مصطفي الزغاوي

سندس محمد حسن

يوسف محمد عبدالفتاح

المشرف: د/ علي عبدالحميد

جامعة الأزهر

كلية الهندسة

قسم الهندسة الكهربية

هندسة الاتصالات والإكترونيات

٦ يوليو ٢٠٢٦